TACOS II DOCUMENTATION. VOLUME IID. TERAIN, PMAP, AND SORTEX. PROGRAMMER/ ANALYST MANUAL (CHAPTER PMAP)

Braddock, Dunn and McDonald, Incorporated

Prepared for:

Army Missile Command

15 May 1974

AD-782 152

BRADDOCK, DUNN AND McDONALD, INC.

1st Natl. Bank Bldg. E. Suite 1717
5301 Central Avenue, NE.
Albuquerque, New Mexico 87108
Phone (505) 266-5711

15 May 1974
BDM/A-37-74-TR-R1
DAAH01-73-C-1150

TACOS II DOCUMENTATION
VOLUME IID
TERAIN, PMAP, AND SORTEV
PROGRAMMER/ANALYST MANUAL
(CHAPTER PMAP)

Submitted to Headquarters, U.S. Army Missile Command, ATTN: AMSMI-DA, Redstone Arsenal, AL 35809.

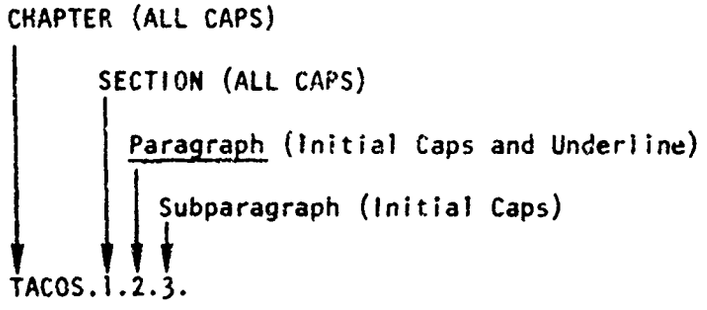
197

FOREWORD

This documentation effort was supported by the United States Army Missile Command under Contract No. DAAH01-73-C-1150, titled "TACOS II Model Documentation."

The documentation is a nine-part, three-volume report. Volume I is the EXECUTIVE SUMMARY, Volumes IIA, B, C, and D are the PROGRAMMER/ANALYST MANUALS for FRAG1, FRAG2, FRAG3, and TERAIN, PMAP and SØRTEV, respectively. Volumes IIIA, B, C, and D are the USER/PLANNER MANUALS for FRAG1, FRAG2, FRAG3, and TERAIN, PMAP and SØRTEV, respectively.

As an aid to the user, a somewhat different form of paragraph and page numbering scheme has been incorporated in this documentation. Each chapter title is the name of that portion of TACOS being discussed. There is one exception to this rule: the first chapter of each of Volumes II and III is a general overview of the TACOS operation. It is titled "Chapter TACOS." The headings within each chapter are of a modified numerical scheme. Except in a few places, the numbering is held to four levels. For each volume, the level designation and accompanying heading name typography is:



Illustrations and figures are numbered consecutively within each section. The chapter and section number are an integral part of the numbering scheme (i.e., Figures FIA.1-1, FIA.1-2, etc.). Page numbering uses essentially the same scheme used for heading designations, however, only three levels are used. The level indicated in the page number corresponds to the major chapter division level on a page. An example would be:

F3C.1.2-1, 2, 3 ... n.

The user needs simply to locate the section of interest in the Table of Contents; he can then turn directly to the appropriate page. A variation on this methodology is used in the documentation for individual programs. The program name being documented or flowcharted is imbedded in the pagination scheme at the third level. In the documentation portion, pages are numbered consecutively with letters of the alphabet i.e.,

F1A.4.MAIN-A, B, C ...

In the flowchart portion, pages are numbered consecutively with numbers corresponding to the flowchart page i.e.,

F1A.4.MAIN-1, 2, 3 ...

This numbering scheme is somewhat nonstandard, but it is designed to afford the reader maximum ease of use.

Principal contributers to this work include: D. E. Edgemon, J. N. Gant, D. R. Jackson, J. S. Nowicki, J. J. Sikora, L. H. Skinner, and R. J. Upham. Project leadership was by R. L. Katz under the directorship of O. V. Fedoroff.

TERAIN, PMAP, AND SORTEV
PROGRAMMER/ANALYST GUIDE
TABLE OF CONTENTS
CHAPTER PMAP

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

CHAPTER: PMAP

PM/P PROGRAMMER/ANALYST GUIDE

P.1 MODULE STRUCTURE

The structure of the PMAP module is defined by specifying the functional flow, a detailed narrative description of each submodel contained in PMAP, a description of inputs required and outputs generated by PMAP and the subroutine call sequence.

P.1.1 Functional Flow

Figure P.1-1 illustrates the overall functional flow of module PMAP. As shown, PMAP reads the input control cards, checks the site data as the sites are input, plots a site map, checks the path data as the paths are input and plots the input paths.

P.1.2 Overlay Structure

PMAP is not overlaid.

P.1.3 Subroutine Call Sequence

The PMAP subroutine tree is diagrammed in Figure P.1-2. The subroutine call sequence is indicated in Figure P.1-3. The name of each called subroutine is indicated in the upper left-hand corner of each box. "Nesting" of subroutine calls is indicated by the nesting of boxes. This figure shows all calls which may be made in the course of a run. This does not include calls to ERR which is called only when an error is detected. Also, the diagram does not include calls to any system supplied utility routines.

P.1.4 PMAP Detailed Narrative Description

The objective of this detailed narrative description is to provide a brief, narrative, non-mathematical, synopsis of each submodel in the subject module. Additionally a secondary objective is to provide some correlation of submodels within this module and some other modules where close correlation exists. A statement-by-statement description of the code contained in each subroutine may be found in the subroutine documentation and associated flow charts in Paragraph P.4.

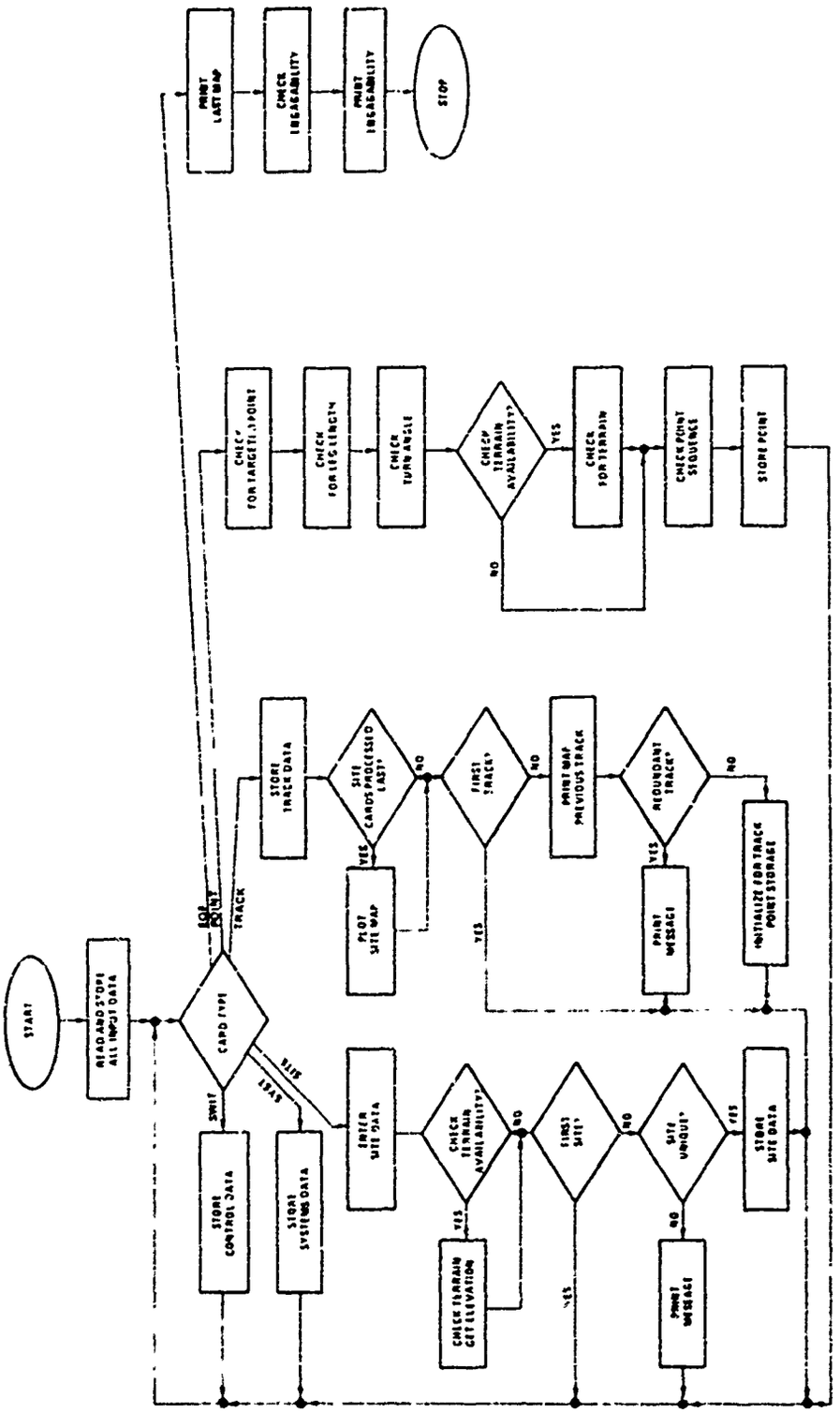


Figure P.1-1. PMAP Functional Flow

2<

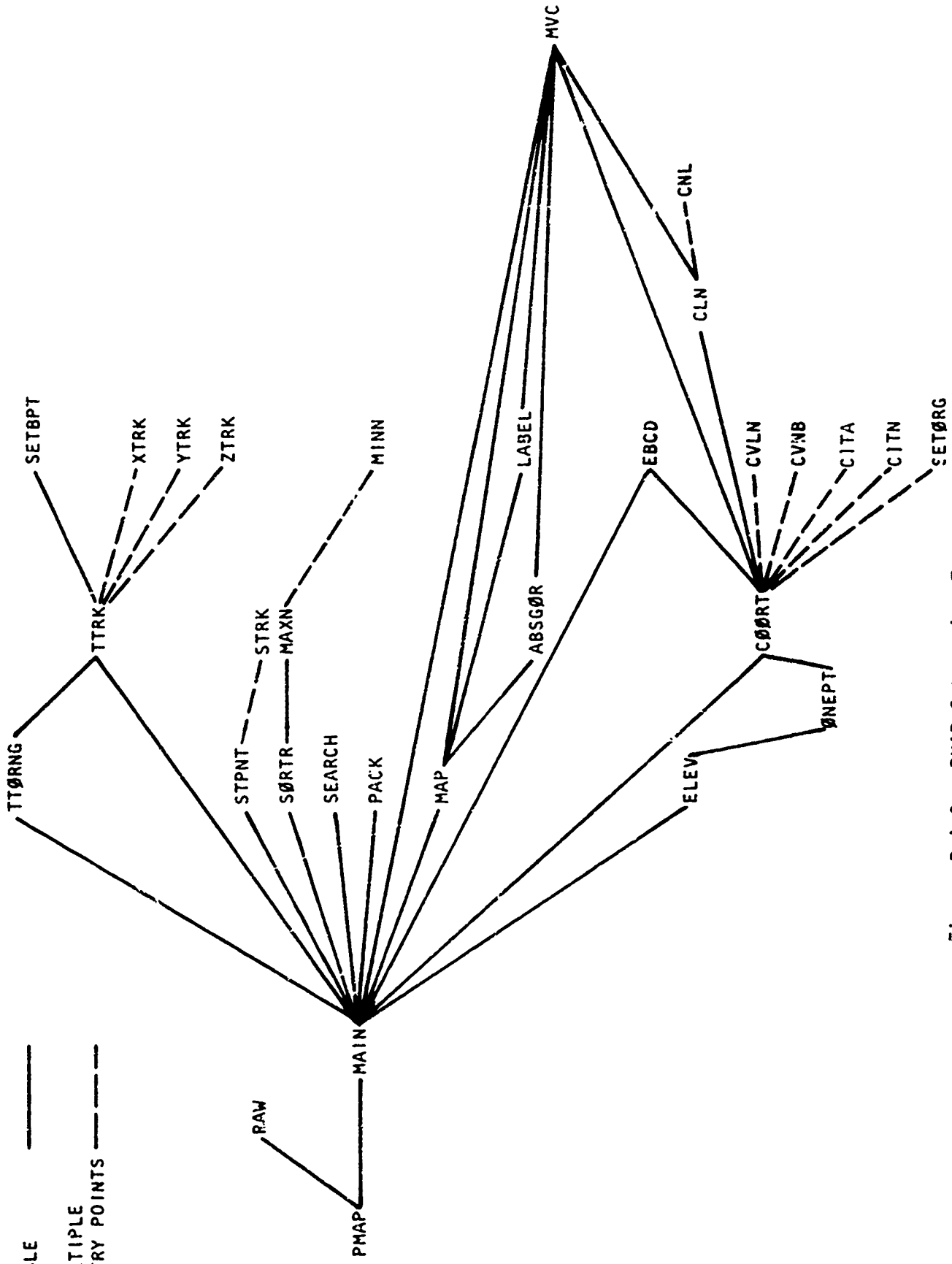


Figure P.1-2. PMAP Subroutine Tree

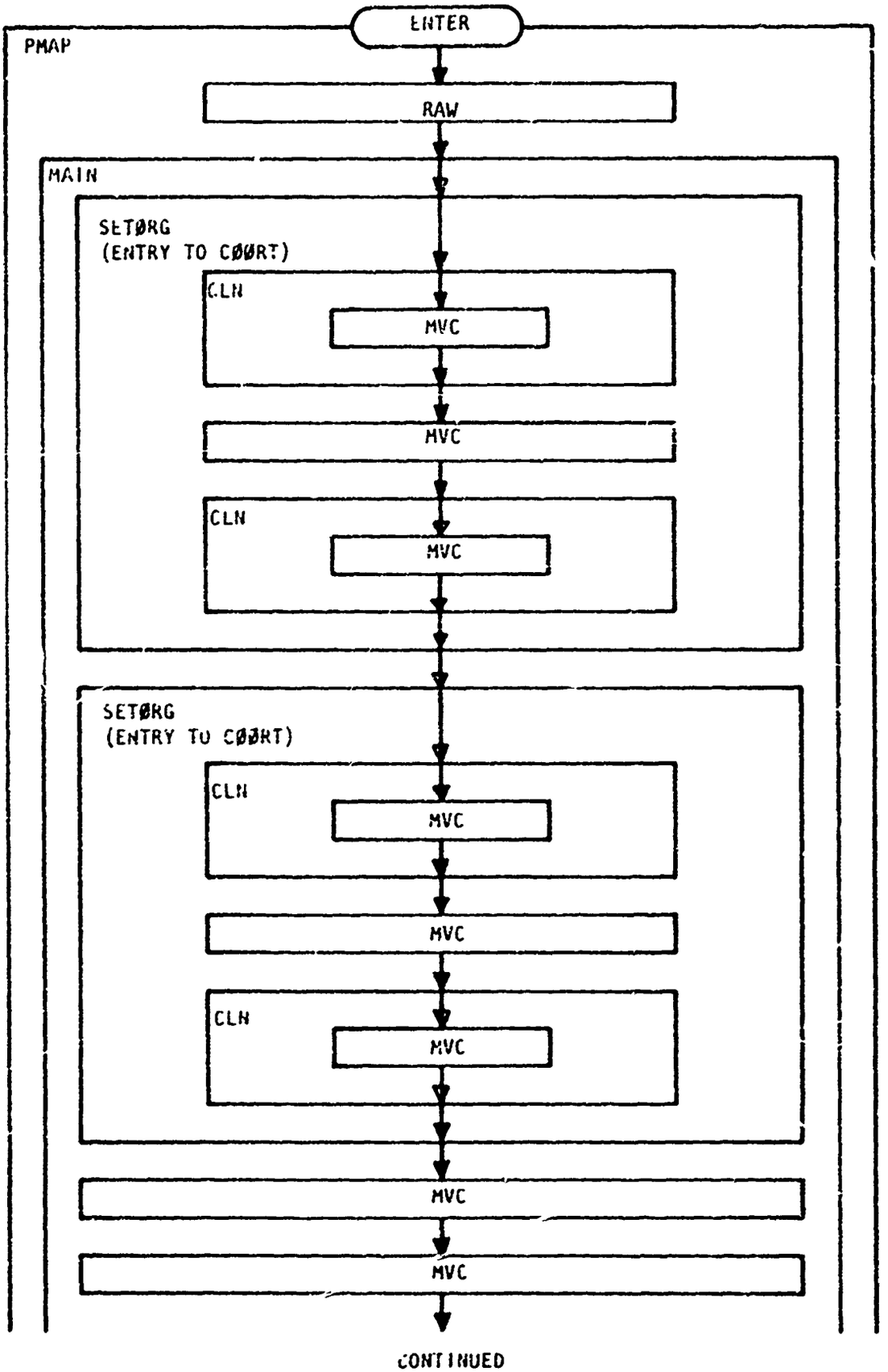


Figure P.1-3. PMAP Subroutine Call Sequence

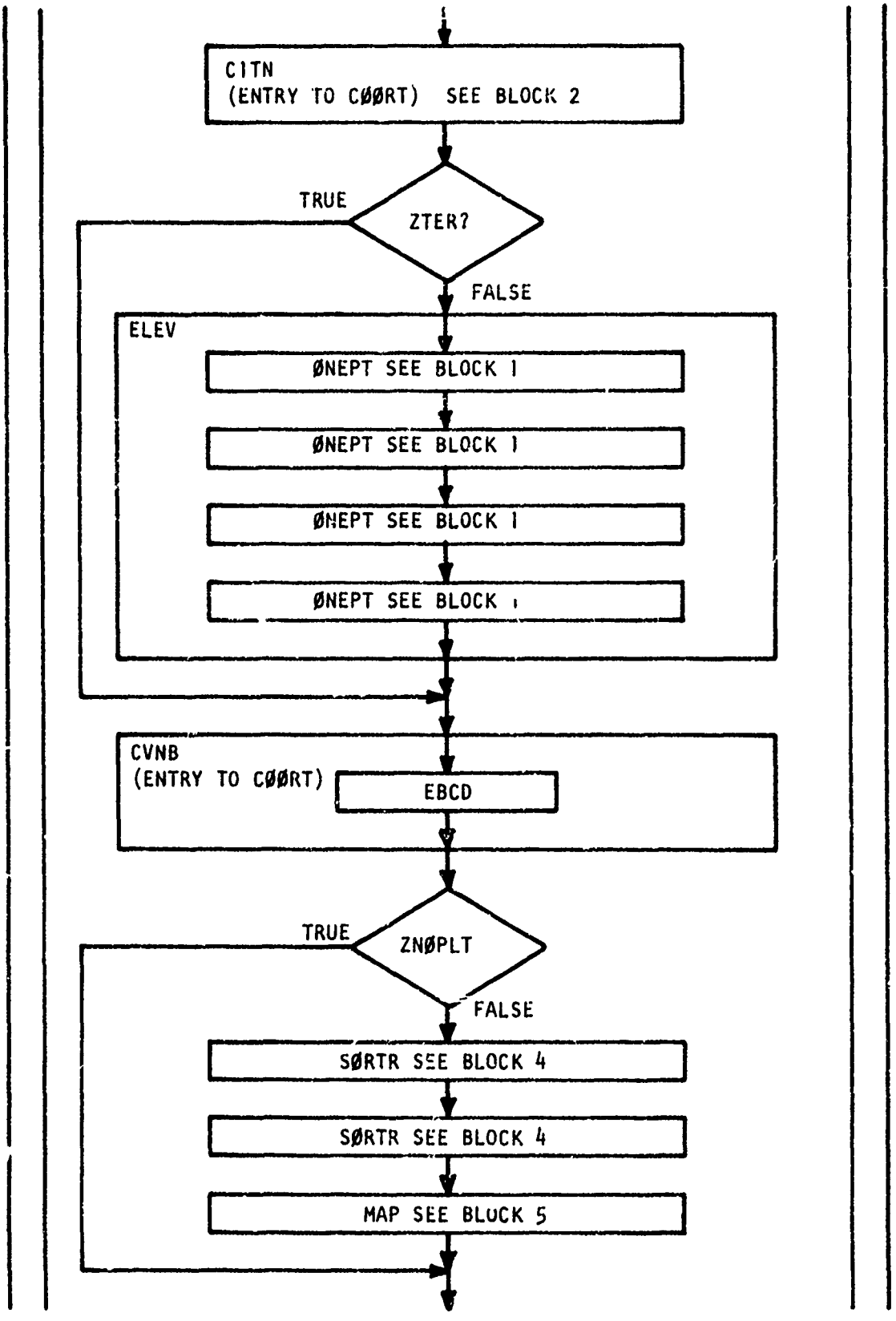


Figure P.1-3. PMAP Subroutine Call Sequence (Continued)

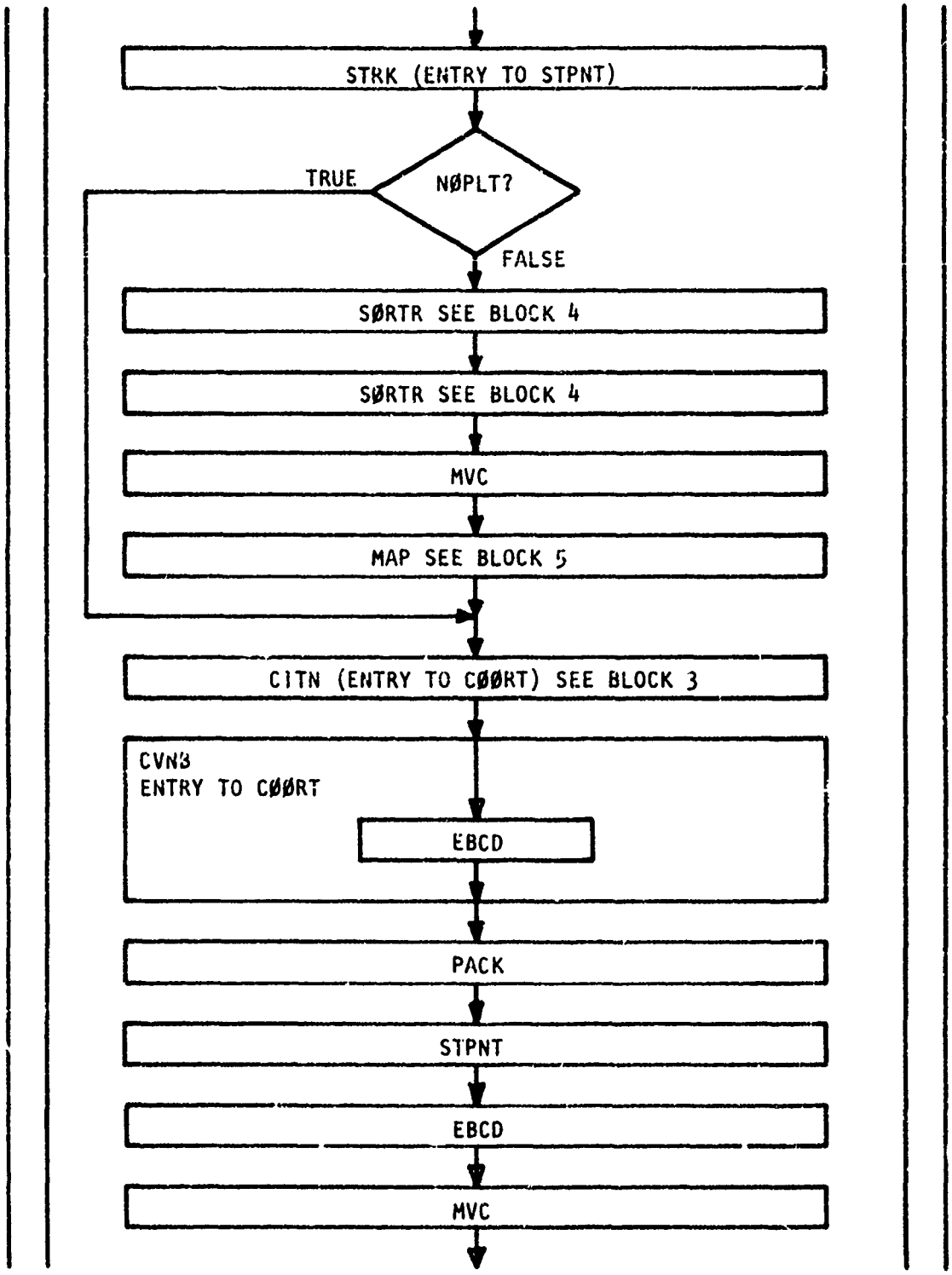


Figure P.1-3. PMAP Subroutine Call Sequence (Continued)

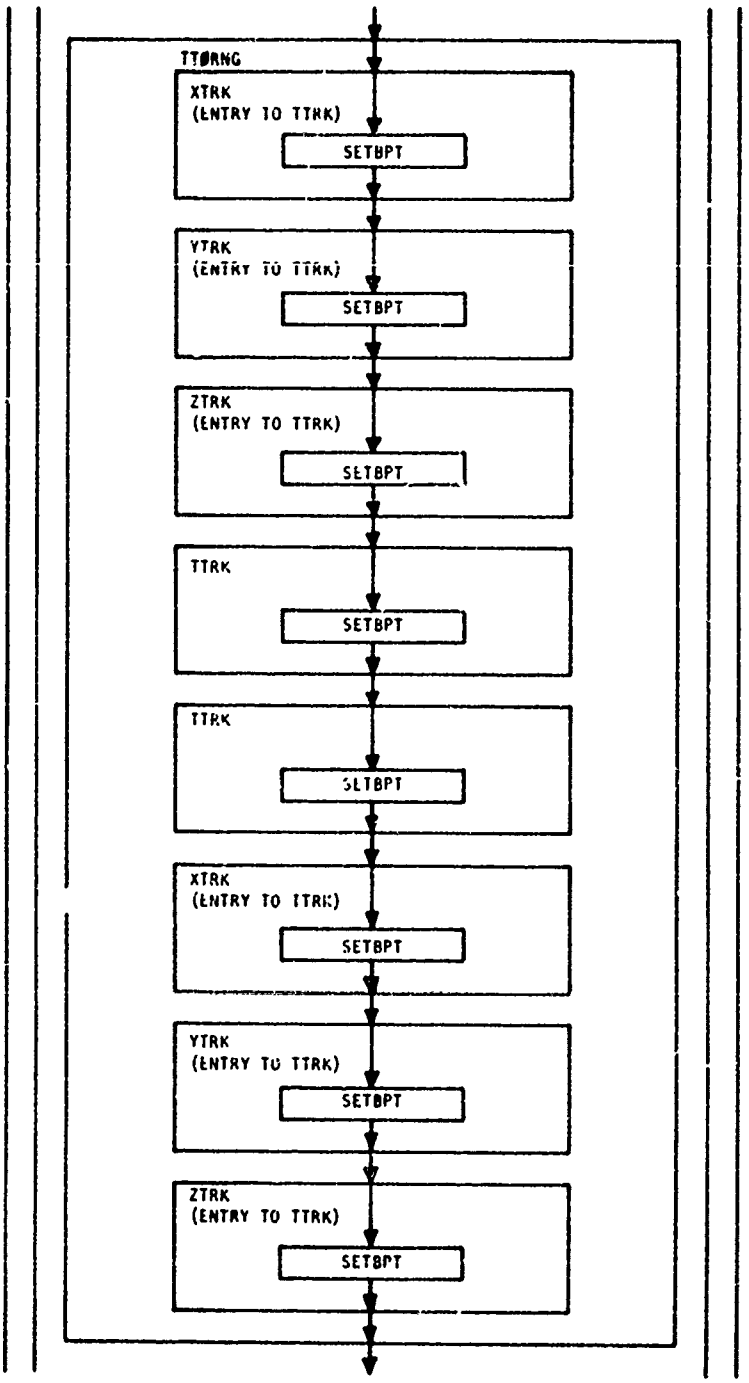


Figure P.1-3. PMAP Subroutine Call Sequence (Continued)

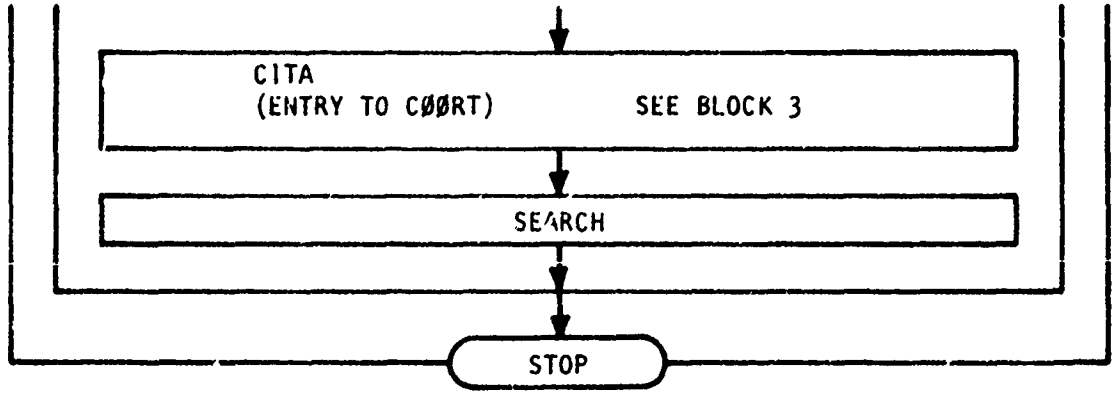
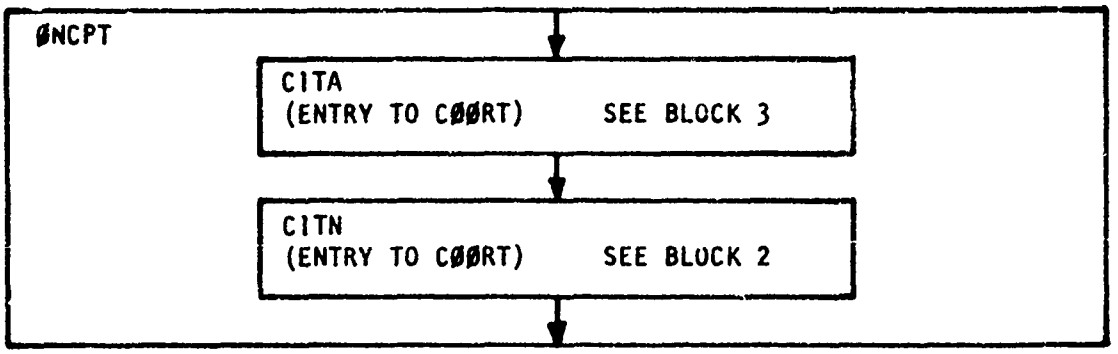


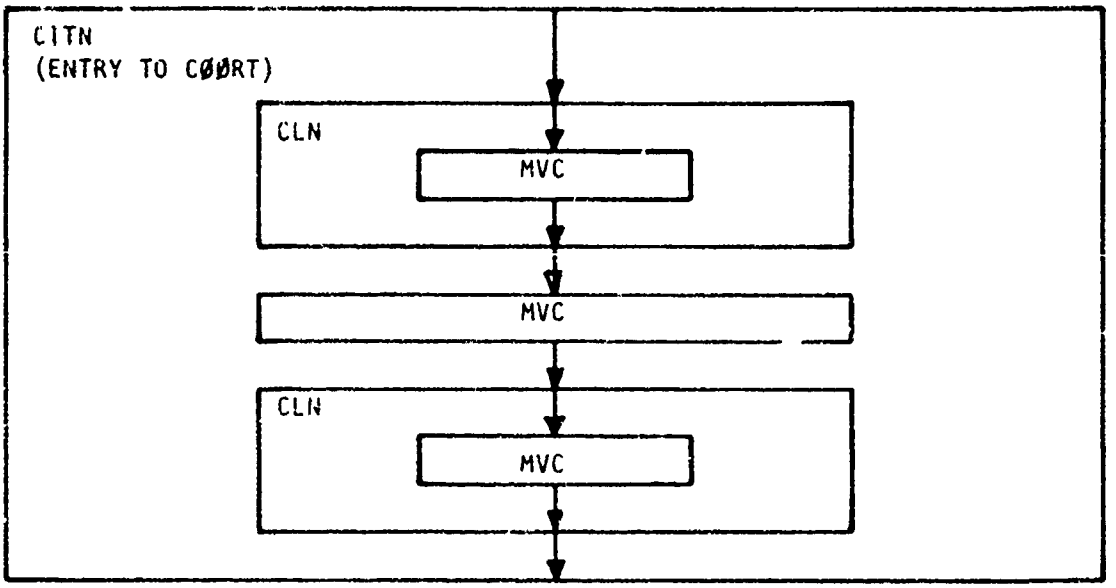
Figure P.1-3. PMAP Subroutine Call Sequence (Continued)

8<

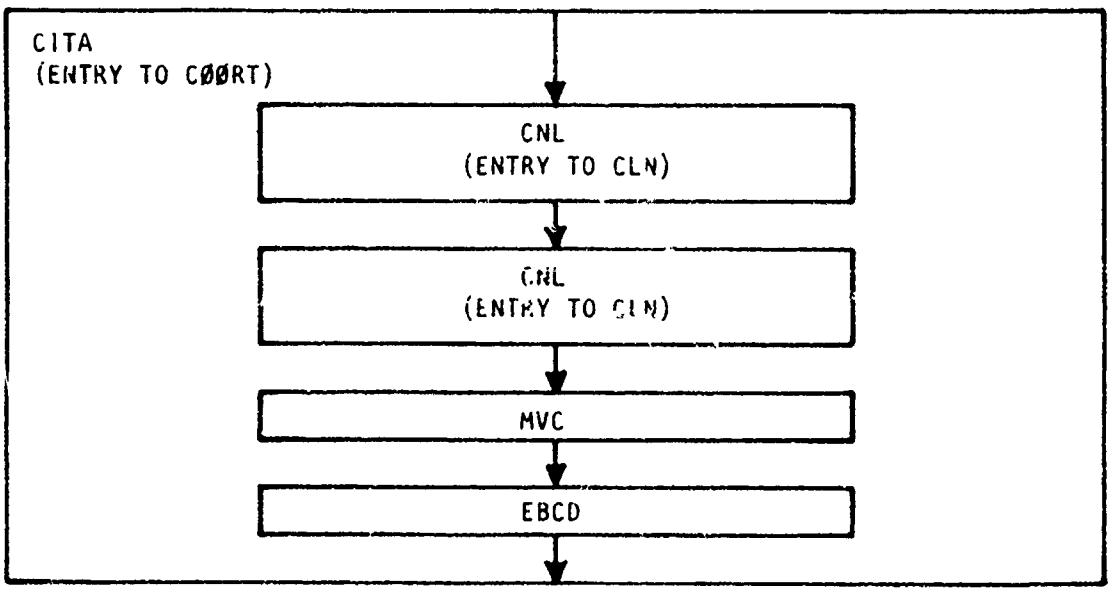


BLOCK 1

Figure P.1-3. PMAP Subroutine Call Sequence (Continued)

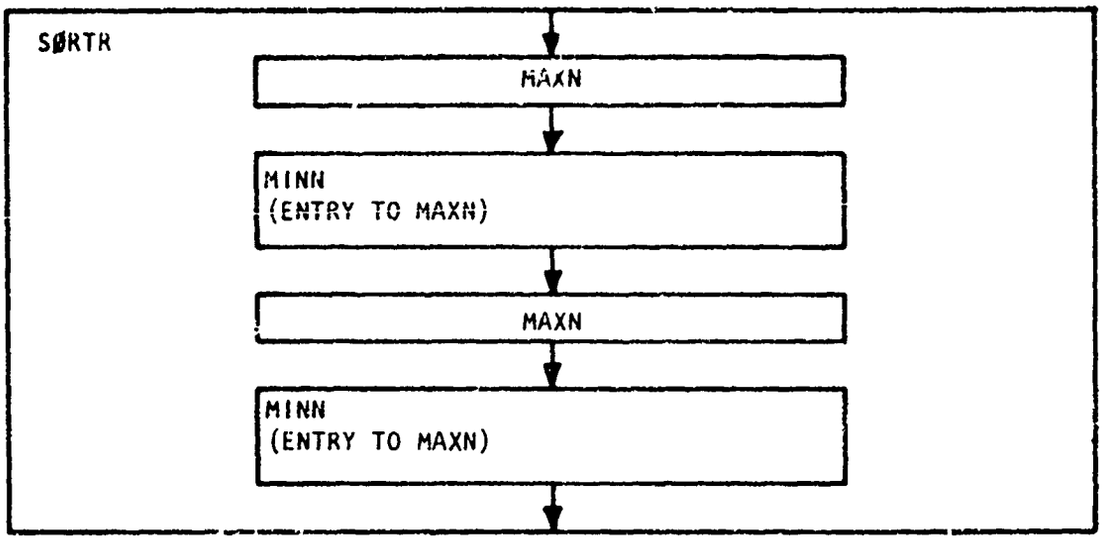


BLOCK 2



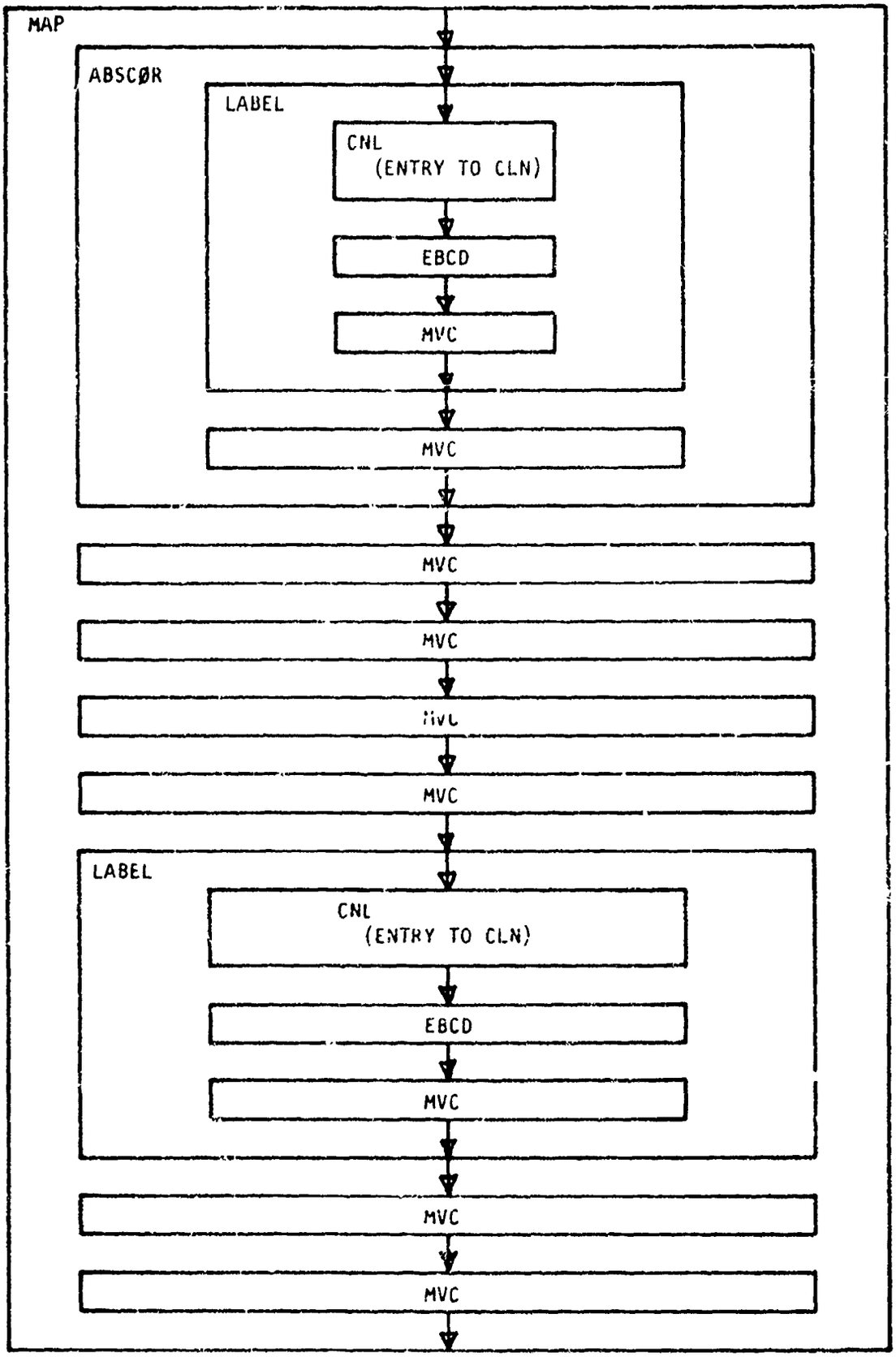
BLOCK 3

Figure P.1-3. PMAP Subroutine Call Sequence (Continued)



BLOCK 4

Figure P.1-3. PMAP Subroutine Call Sequence (Continued)



BLOCK 5

Figure P.1-3. PMAP Subroutine Call Sequence (Concluded)

P.1.4.1 Sequence of Operation

As shown by Figure P.1-1 (PMAP Functional Flow), the sequence of operation is basically to read the input systems and site cards, perform the specified site data checks and site plotting, read the input path data, perform the specified path checks and plot the paths, and determine site/path engageability.

P.1.4.2 Terrain Data Base

PMAP utilizes the identical real-world terrain data base as does FRAGIA and FRAGIB. This data base is used to check for terrain availability at the site locations and path points and to check the input site altitudes.

P.1.4.3 Site Data Checking

As the SITE cards are input to PMAP, they are checked for redundancies, such as duplicate site cards, sites with duplicate names and sites with identical locations. If requested, each site is also checked for terrain data related errors. Anomalies are entered in the site description table.

P.1.4.4 Path Data Checking

As the path data are read, each path is checked to determine if it contains an unusually long path leg and each turning point is checked for an excessive turn angle. Anomalies are entered in the path description table.

P.1.4.5 Targeted Path Identification

PMAP identifies paths which are targeted against specific sites by matching analyst specified path point identifiers with site identifiers. Targeted path points are identified in the path description table.

P.1.4.5 Path Engageability Identification

For each site, PMAP determines the paths which pass within the site range of interest. These paths are considered engageable by this site and are entered for that site in the site engageability table.

P.1.4.6 Map Plotting

A PMAP plot consists of printing the site identifiers at the site locations and the path point sequence numbers at the point locations on a UTM grid. One plot is printed for all the site locations and one plot for each of the input paths.

P.2 INPUT RECORD FORMATS

P.2.1 Digitized Terrain Data Tape

The digitized terrain data tape utilized by PMAP has the identical format as the terrain data tape utilized by FRAGIA. For the detailed tape format see FIA.2, Volume II A FRAGI Programmer/Analyst Manual.

P.2.2 Control Cards

In order to perform the preliminary data checks and plot the site and path maps, PMAP expects the following parameter types:

- Terrain file specification
- Control logicals
- Systems data
- Site specifications
- Path specifications

A more detailed explanation of the card type inputs is found in Volume III D TERRAIN, PMAP, SORTEV User/Planner Manual, Paragraph P.2.

P.3 MATHEMATICAL/LOGICAL STRUCTURE

PMAP utilizes the following five submodels: Site Data Checking, Path Data Checking, Targeted Path Identification, Path Engageability Identification, and Map Plotting.

P.3.1 Site Data Checking

P.3.1.1 Purpose;

This submodel identifies probable errors in site specification.

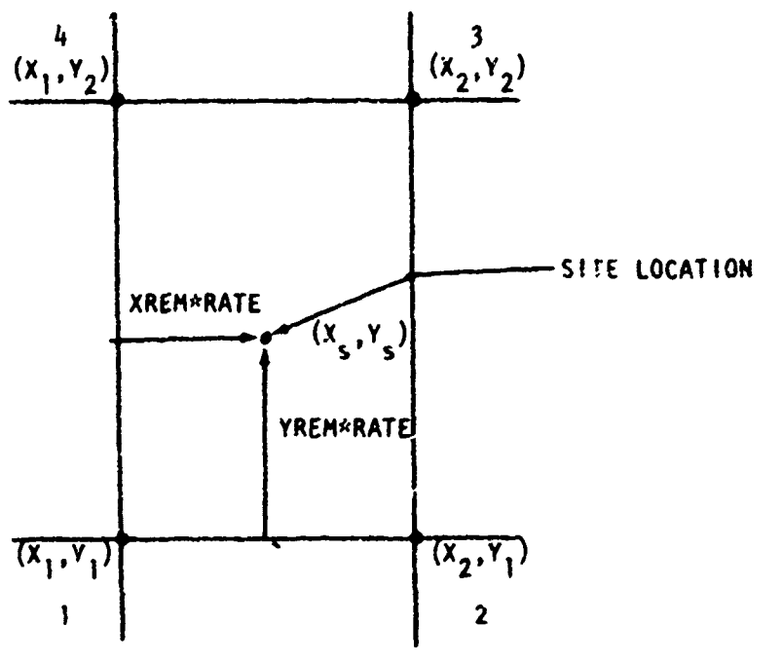
P.3.1.2 Algorithm/Function Definition:

Checks are performed on a card by card basis.

P.3.1.3 Rationale for Derivation:

As each SITE card is read, the terrain data directory is accessed to determine if the terrain file contains the grid square containing the site location. If terrain data are available for this site location, the terrain data are used to interpolate for the terrain elevation at the site location. If terrain is not available at the site location or the site altitude specified is less than the terrain elevation, an informative message is printed in the site description table.

The site elevation is determined by bilinear interpolation of the four closest terrain data sample points. The procedure is as follows:



$$X_1 = RATE * \left[\frac{X_s}{RATE}\right]$$

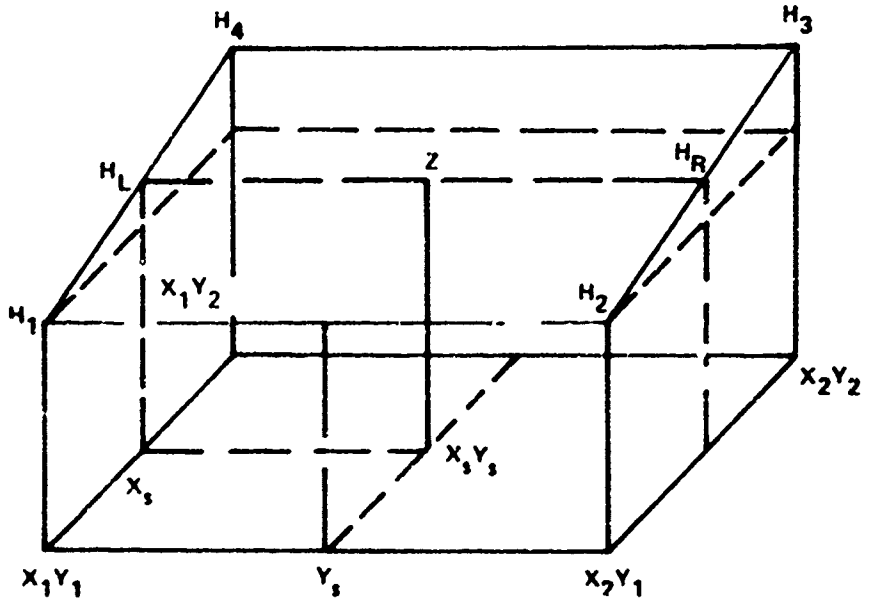
$$Y_1 = RATE * \left[\frac{Y_s}{RATE}\right]$$

$$X_2 = RATE + X_2$$

$$Y_2 = RATE + Y_1$$

$$XREM = \left\{ X_s - \left[\frac{X_s}{RATE}\right] \right\} / RATE$$

$$YREM = \left\{ Y_s - \left[\frac{Y_s}{RATE}\right] \right\} / RATE$$



$$H_L = (H_4 - H_1) * YREM + H_1$$

$$H_R = (H_3 - H_2) * YREM + H_2$$

$$Z = (H_R - H_L) * YREM + H_L$$

Where: RATE is the lateral data rate

[K] is "the integer part of K"

A card by card check is then performed to determine if a site card has been previously processed whose site identifier, location, or both identifier and location are identical with the card under consideration. If any of these duplications are detected, an "error" message is printed in the site description table.

P.3.1.4 Parameter Definitions: None

P.3.1.5 Assumptions and Rationale: None

P.3.1.6 Data Edit Requirements and Manual Procedures: None

P.3.2 Path Data Checking

P.3.2.1 Purpose

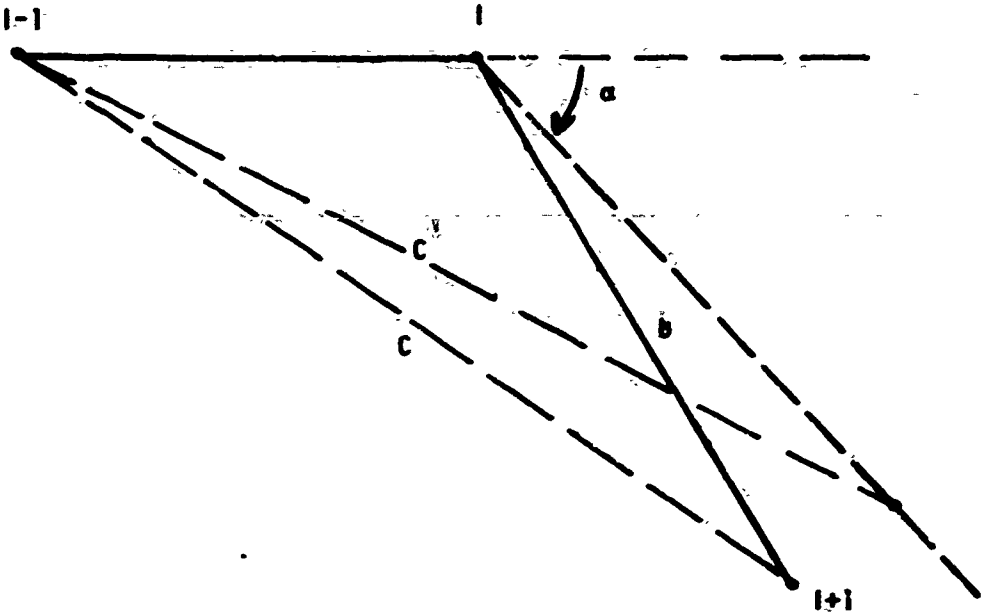
This submodel identifies probable errors in path specification.

P.3.2.2 Algorithm/Function Definition

Checks are performed on a card by card basis.

P.3.2.3 Rationale for Derivation

As each 'P$INT' card is read, the two letter grid square designator for the point location is matched against the valid designators from the terrain file directory to determine if this point lies on available terrain. Upon entry of the second and all succeeding cards, the leg length of the preceding leg is calculated and this length checked to determine if it is longer than the maximum specified. Upon entry of the third and all succeeding points, the path turn angle is tested to determine if the maximum turn angle for the previously input point has exceeded the maximum expected turn angle. The test for maximum turn angle is developed as follows:



$$A^2 = (X_I - X_{I-1})^2 + (Y_I - Y_{I-1})^2$$

$$B^2 = (X_{I+1} - X_I)^2 + (Y_{I+1} - Y_I)^2$$

$$C^2 = (X_{I+1} - X_{I-1})^2 + (Y_{I+1} - Y_{I-1})^2$$

$$C'^2 = A^2 + B^2 + 2.*A*B*COS\alpha$$

Where α = Maximum turning angle

The maximum turning rate is exceeded if

$$C'^2 > C^2$$

As the path is entered, the path points' easting, northing, altitude and checkpoint code are stored for later use as shown in Figure P.3-1.

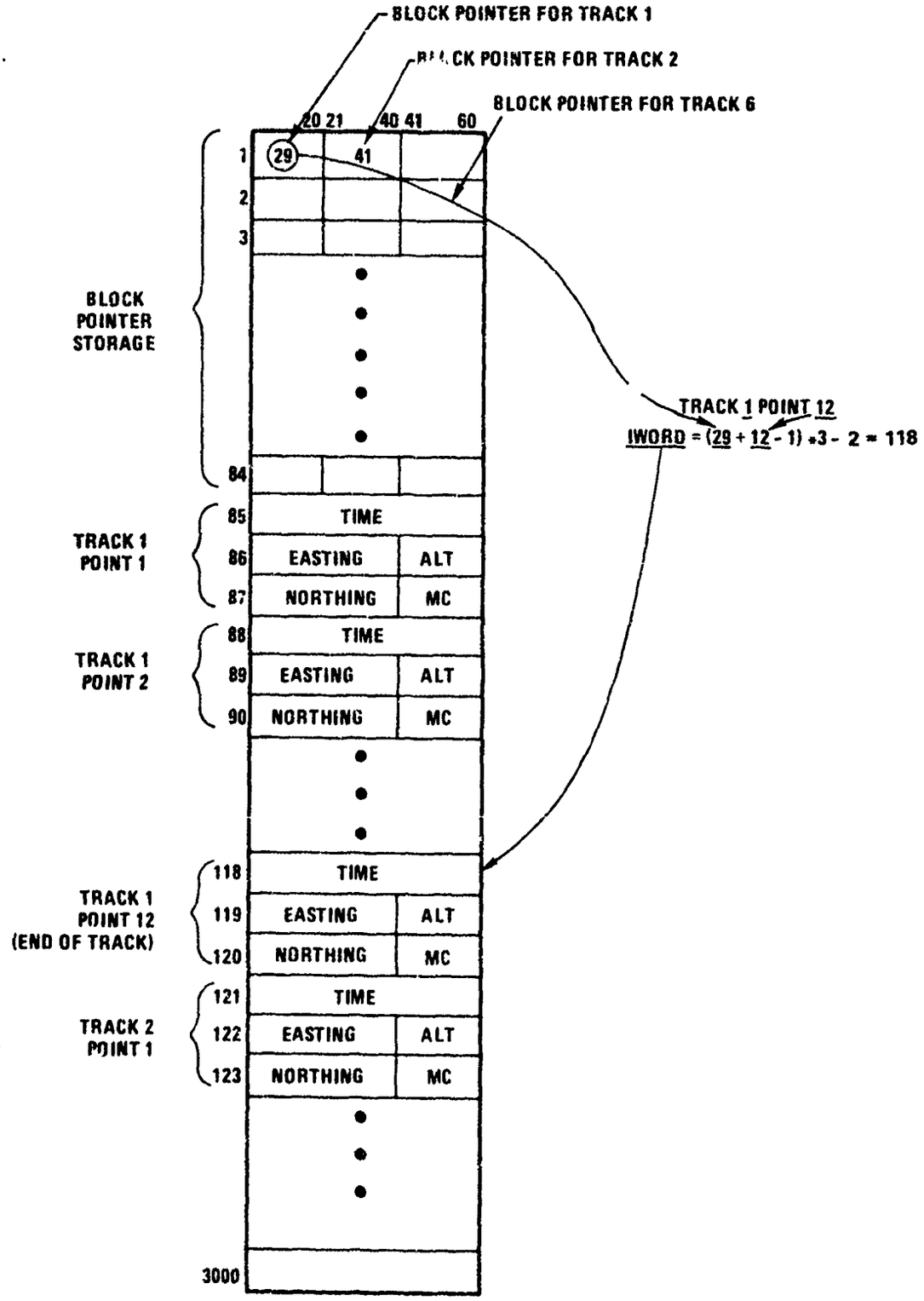


Figure P.3-1 Path Point Storage

P.3.2.4 Parameter Definitions: None

P.3.2.5 Assumptions and Limitations: None

P.3.2.6 Data Edit Requirements and Manual Procedures: None

P.3.3 Targeted Path Identification

P.3.3.1 Purpose

This identifies explicitly the path points which are targeted for a specified site.

P.3.3.2 Algorithm/Function Definition

A path is considered to be targeted for a site if an input path point location is within 25 meters ground range of that site location.

P.3.3.3 Rationale for Derivations

As each path point is read, the point identifiers are compared with each of the site identifiers to determine if this point is a candidate for targeting against any of the stored sites. If a match is determined, the range from that site to the path point is calculated. This range is then tested and if it is less than 25 meters, this path is targeted against this site.

P.3.3.4 Parameter Definitions: None

P.3.3.5 Assumptions and Rationale: None

P.3.3.6 Data Edit Requirements and Manual Procedures:

If a path point is to be considered for targeting, its identifier must match the site identifier under consideration.

P.3.4 Path Engageability Identification

P.3.4.1 Purpose

This submodel determines which paths intersect each site engagement volume.

P.3.4.2 Algorithm/Function Definitions

Whenever a vehicle passes within a site's range of interest, it is considered engageable.

P.3.4.3 Rationale for Derivation

For each site/path combination, the vehicle path is checked, path leg by path leg, to determine if this path leg intersects this site's engagement volume. Each path leg is tested utilizing the rationale developed in Figure P.3-2. First, the square of the distance between the path leg end points (B^2), the square of the range from the site to path point i (C^2) and the square of the range from the site to the path point i+1 (A^2) are calculated by summing the squares of the differential easting, northing and altitudes. The cosine of the angle γ is then determined by the law of cosines. The side B' of triangle A, B', R' is then determined as shown. If it is then determined that R' is less than the range of interest, the path intersects this site's engagement volume.

P.3.4.4 Parameter Definitions:

P.3.4.4.1 Range of Interest

That range within which it is possible for an air defense site to detect an airborne vehicle.

P.3.4.4.1.1 Source: System Description

P.3.4.4.1.2 Representative Values:

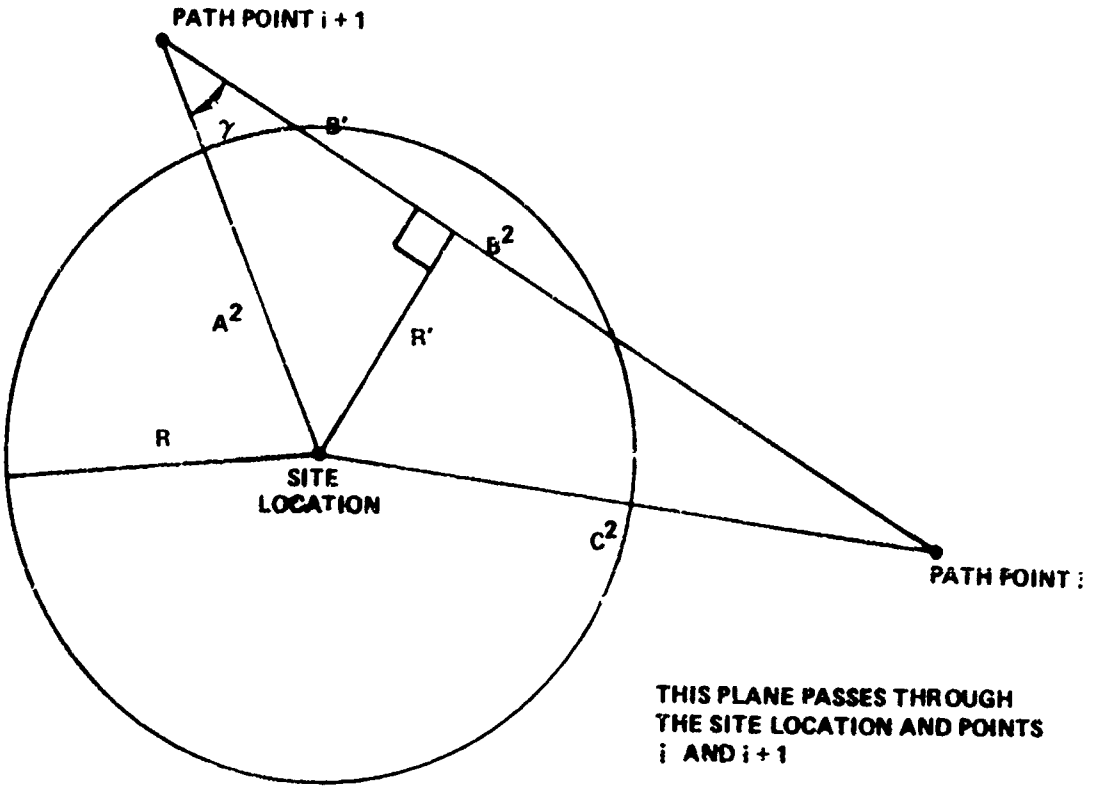
35000,78000 meters.

P.3.4.4.1.3 Ranges:

1000,000 meters

P.3.4.5 Assumptions and Rationale: None

P.3.4.6 Data Edit Requirements and Manual Procedures: None



$\cos\gamma = (A^2 + B^2 - C^2)/(2. * A * B)$

$R' = A\sin\gamma$

PATH ENGAGEABLE IF $R' < R$

Figure P.3-1. Path Engageability

P.3.5 Map Plotting

P.3.5.1 Purpose:

Provide graphical displays of site locations and vehicle paths.

P.3.5.2 Algorithm/Function Definitions:

PMAP plots consists of printing the site identifiers at the site locations and the path point sequence numbers at the point locations on a UTM grid.

P.3.5.3 Rationale for Derivation:

Once the points to be plotted, either site locations or path points, have been determined, the plot algorithm begins by first sorting the points and restoring them in the order of descending northing. Next, the northern-most, southern-most and western-most points are determined. The map plotting algorithm then uses the northern-most point to set the map northern boundary, the western-most point to set the weastern boundary and the southern most-point to set the southern boundary. The width of the strips to be plotted is then determined from the plot scale and page width. The easting of the eastern boundary of the first strip to be plotted is then determined by subtracting the strip width from the easting of the western boundary. This north-south strip is then printed, line by line, with the UTM grid and point identifiers printed as required. If all the points under consideration did not lie on this strip, the algorithm moves one strip east and prints the next north-south strip. This process continues until all the points under consideration have been plotted.

P.3.5.4 Parameter Definitions: None

P.3.5.5 Assumptions and Rationale: None

P.3.5 6 Data Edit Requirements and Manual Procedures: None

P.4 PROGRAM LOGIC

P.4.1 Chapter Arrangement

The following chapter presents detailed program information about the routines which comprise PMAP.

The routines are presented in alphabetical order arranged in two parts. For each routine the first part consists of a 13 point documentation and the second part consists of an annotated logic flowchart.

P.4.2 Flowchart Conventions

Off-page connectors are of two types: (1) to following page and, (2) to another page. A Type 1 connector occurs only at the end of a page and contains the number of the page to which it is connected. The same number appears in a connector box at the top of its associated page. The Type 2 connector consists of two boxes. The upper box contains the page number to which the connector points. The lower box contains the alphanumeric designator, on that page, to which the connector points. For example, on Page 2 of subroutine TØDD in SØRTEV, appears an off-page connector labeled $\frac{3}{A1}$. This means that this connector points to a connector labeled A1 which is on Page 3. Additionally, the 1 indicates it is the first connector referenced. Halfway down Page 3 is another connector labeled A2. These connectors are labeled A1 through A9, B1 through B9, etc. Additionally, each Type 2 on-page connector is accompanied by a label of the page (or pages) number that references this connector. Connections between the first and last statements of a DO loop are indicated by a dashed line. If these statements are on separate pages, the dashed line from the first statement terminates with

a pointer to the appropriated page. The dashed line for the last statement on the other page terminates with a pointer back to the originating page.

Each FORTRAN statement number appears to the upper left of its appropriate logic flow box.

The subroutines comprising PMAP are:

ABSCØR	MAXN	SØRTR
CLN	MVC	STPNT
CØØRT	MYTIME	THYME
CVNBM	ØNEPT	TTØRNG
EBCD	PACK	TTRK
ELEV	PMAP	ZCKPTI
ERR	RAW	
JTRK	SEARCH	
LABEL	SETBPT	
MAIN		
MAP		

1. NAME: ABSCØR

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Fill the print buffer with the abscissa coordinate characters to be printed.

5. ASSUMPTIONS AND LIMITATIONS: The number of labels must fit on the page at the selected scale factor.

6. ERROR RETURNS: A call ERR and a RETURN A1 will be executed if the following errors are detected:

   A. The abscissa coordinate values exceed 132 spaces at the specified spacing.

   B. An illegal grid letter was returned by CNL.

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL ABSCØR(ISTART, ISPC, N), RETURNS (A1)

   ISTART - Integer, the starting position for the abscissa coordinates in the print array.

   ISPC - The number of blanks between coordinate labels.

   N - Number of labels to be printed across the page.

   A1 - Exit taken if one of the conditions listed under 6 (above) occurs.

8. PROGRAM CALLING THIS PROGRAM: MAP

9. COMMON VARIABLES USED:

   A. ZZCØRD - DELTA, ESTEDG, IØRBUF, WSTEDG

   B. ZZCVI - XØRG

10. COMMON VARIABLES TO BE SET:

    A. ZZCØRD - DELTA, ESTEDG, WSTEDG,

    B. ZZCV1 - XØRG

11. COMMON VARIABLES CHANGED:

    A. ZZCØRD - IØRBUF

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

    A. SUBROUTINES

        (1) CALL ERR (NAME, LAST, NSTAT) - Prints an error message indicating name of subroutine and statement number where error occurred.

            NAME - Name of subroutine where error was detected.

            LAST - End address of the storage area to be dumped from subroutine NAME.

            NSTAT - Statement number in subroutine NAME where error occurred.

        (2) CALL LABEL (M, X, XØRG, ID), RETURNS (A1)

            M - Character, contains the one letter, three digit grid label for the line X.

            X - Relative easting (ID=E) or northing (ID=N) for which the UTM grid letter is desired.

            XØRG - Displacement (meters) of the origin from the UTM reference point (AA).

            ID - Character, E indicates easting of strip is desired N indicates northing of line is desired.

            Ai - Control is transferred to this statement when an illegal grid letter is detected by CLN.

(3) CALL MVC (ICNT, INPTR, ZINADD, IØPTR, ZIØADD) - Move characters from the word ZINADD into the word ZIØADD.

ICNT - Integer, number of characters to be moved from ZINADD to ZIØADD.

INPTR - Integer, index of the first character to be moved from ZINADD.

ZINADD - Character, 2 word array (20 character) from which characters will be moved.

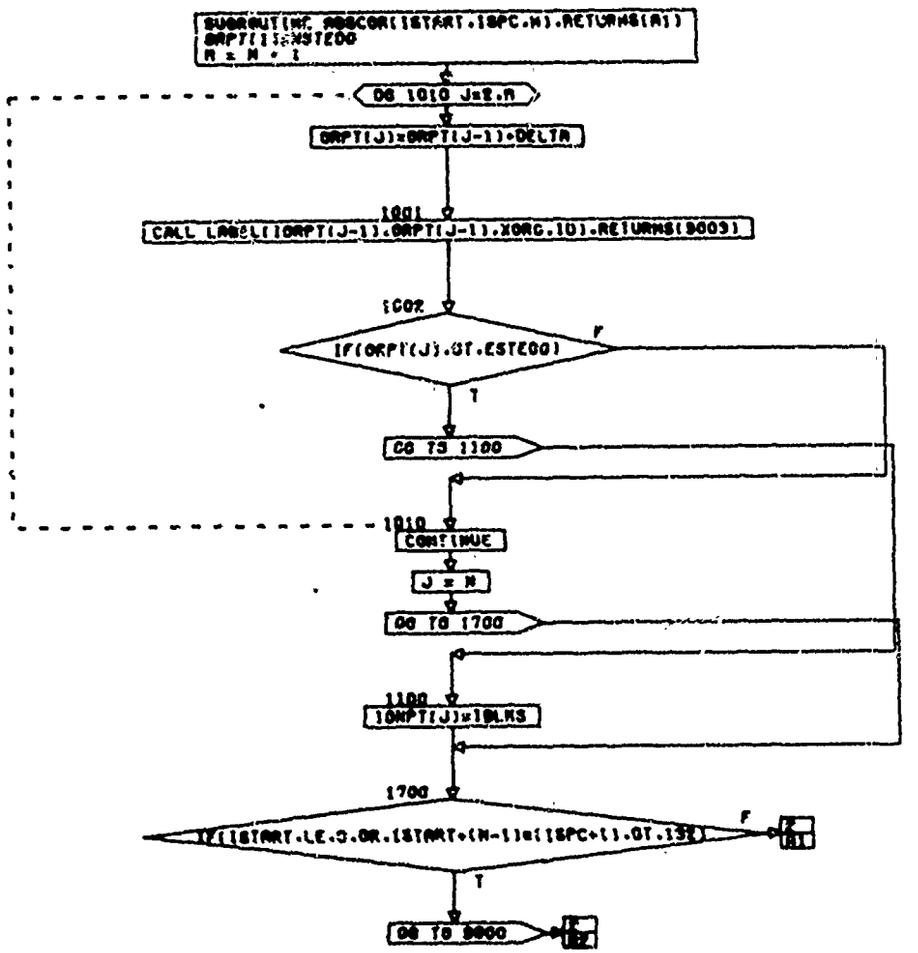
IØPTR - Integer, index of the location in ZIØADD where the first character will be moved.

ZIØADD - Character, 2 word array (20 character) from which the characters will be moved.

13. NOTES ON METHODOLOGY: Upon entry to ABSCØR, ØPTR(1) is set to the easting of the western edge (WSTEDG) of the map and M is set to the number of labels to be printed plus one. A DO loop is then entered to generate the coordinate abscissa labels (1 letter, 3 digits). LABEL is used to generate the labels (IØRPT) at DELTA meters (intervals), starting at the western edge. This loop generates labels until the eastern edge (ESVEDG) of map is passed or the maximum number of labels (N) have been found. If N labels are generated before the eastern edge is passed, J, the number of labels actually generated, is set to N and control transfers to statement 1700. If the eastern edge is passed, control transfers to statement 1100 where the last label is set to a field of blanks. At statement 1700 the starting position (ISTART) in the print array and the last position in the print array are tested. If the start position is less than one or the last position is greater than 132, control is transferred to statement 9000 for error processing.

M is then set to J minus one and a DO loop executed which places blank fill in all the entries of the array IØRBUF. A major DO loop is then entered which moves the 4 characters (one letter, 3 digit) representing the abscissa coordinate label into the buffer IØRBUF. The counters MØRE and LC keep track of the location in IØRBUF where the characters are to be stored. An inner DO loop moves the characters for each abscissa into the buffer IØRBUF. When all four characters for each abscissa coordinate label have been stored in IØRBUF, control transfers to the calling program.

Upon detection of an error in ABSCØR, control is transferred to the appropriate error statement where a call is made to ERR, which prints an error message. A RETURN A1 is executed and control transferred to the calling program.



Store easting of western edge
Set number of labels

DO over abscissa labels

Calculate position

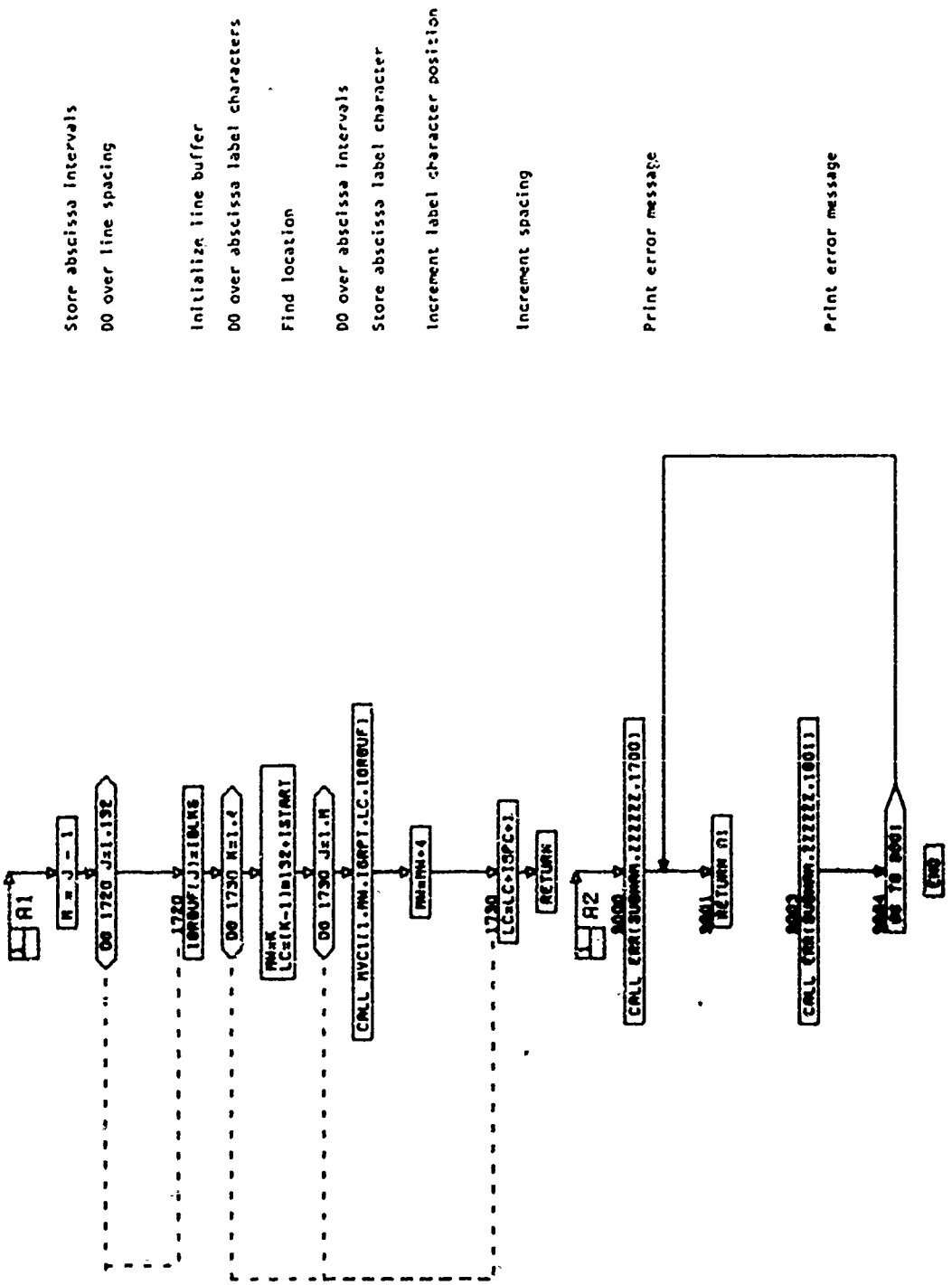
Store label

Have all the abscissa labels been generated?

Store number of labels

Set last label blank

Are any of these labels off the page?


Store abscissa intervals
DO over line spacing
Initialize line buffer
DO over abscissa label characters
Find location
DO over abscissa intervals
Store abscissa label character
Increment label character position
Increment spacing
Print error message
Print error message
A1
A2
RETURN
RETURN A1
END

1. NAME: CLN (other entry points: CNL)

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Calculate the number of 100 km squares between a reference point and given grid square designator. Given the number of squares between the reference and a designator, find the appropriate grid square designator.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: A RETURN A1 will be executed if an illegal grid square designator is found.

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL CLN (LL, N, D), RETURNS (A1)

   entry point CALL CNL (LL, N, D), RETURNS (A1)

   LL - Integer, two letter grid square designator.

   N - Integer, number of 100 x 100 km grid squares between input point and the reference point (location AA).

   D - Indicates whether the squares counted are easting (E) or northing (N).

   A1 - Exit is taken when an illegal grid square designator is detected.

8. PROGRAMS CALLING THIS PROGRAM:

   Entry point CLN: CØØRT

   Entry point CNL: CØØRT, LABEL

9. COMMON VARIABLES USED: None

10. COMMON VARIABLES SET: None

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTIONS OF LINKAGES:

A. SUBROUTINES

(1) CALL MVC (ICNT, INPTR, ZINADD, IØPTR, ZIØADD) - Move characters from the word ZINADD into the word ZIØADD.

ICNT - Integer, number of characters to be moved from ZINADD to ZIØADD.

INPTR - Integer, index of the first character to be moved from ZINADD.

ZINADD - Character, 2 word array (20 character) from which characters will be moved.

IØPTR - Integer, index of the location in ZIØADD where the first character will be moved.

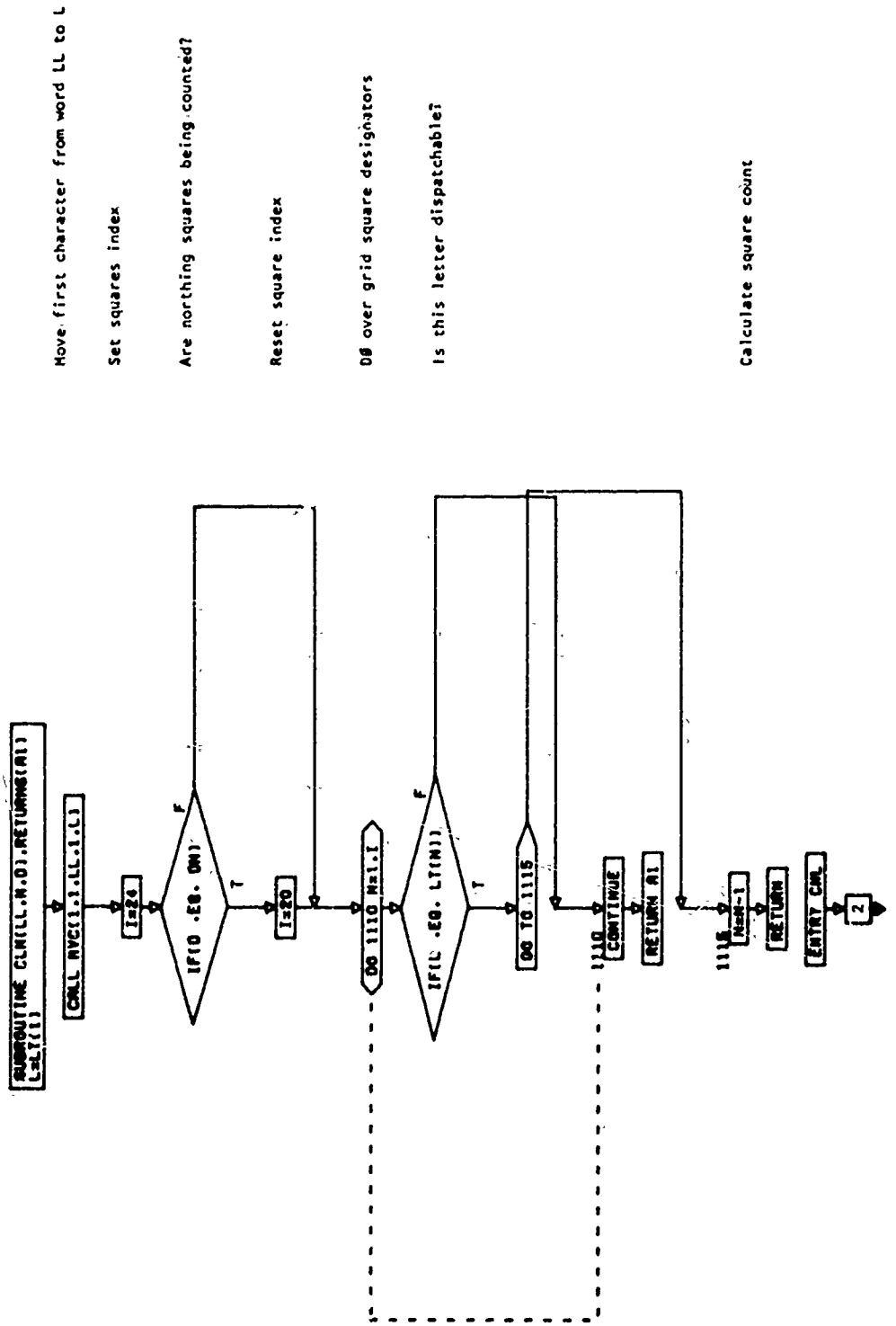
ZIØADD - Character, 2 word array (20 character) from which the characters will be moved

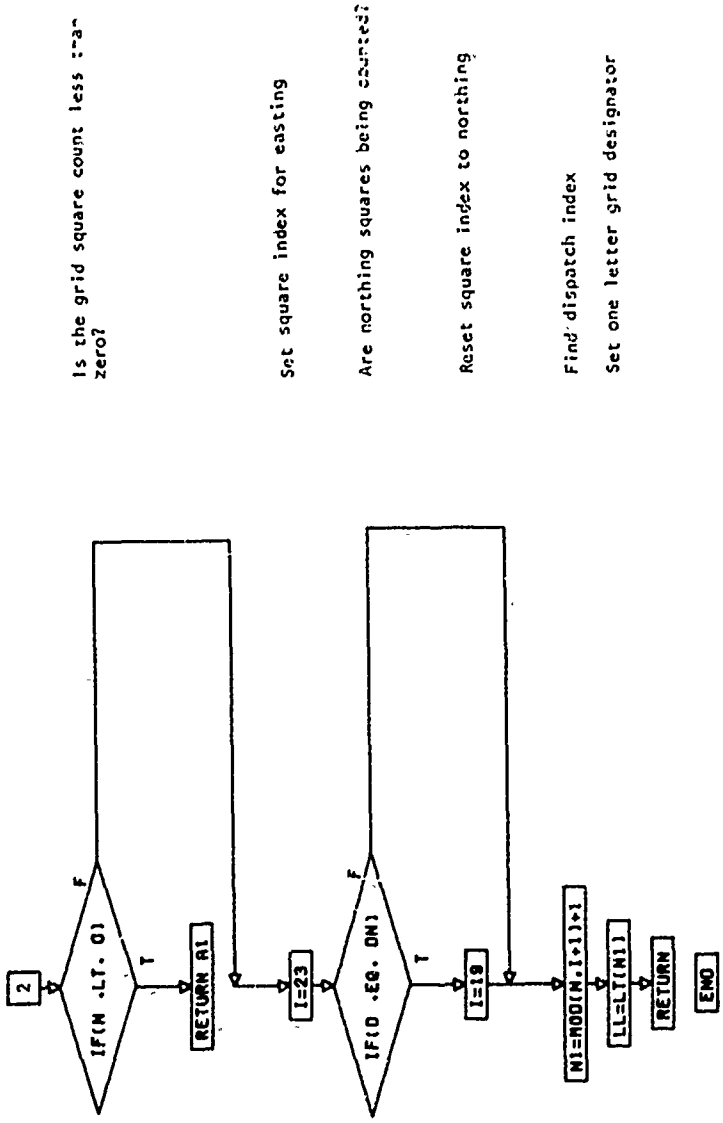
13. NOTES ON METHODOLOGY: CLN provides the logic necessary to count the number of 100 km grid squares between a reference location AA and a specified grid square corner. By ENTRY CNL the one letter grid square designator can be determined from a count of the number of squares between the reference point and the grid square corner.

Upon entry, CLN sets L equal to the one letter grid square designator of the reference point (A). Next MVC is called. MVC sets L equal to the first letter of the grid square designator LL. I, the number of legal grid square letters is next determined. If D is equal 1HE, an easting count is desired and I is set to the number of legal easting points (24). If D is equal 1HN, a northing count is desired and I is set to the number of legal northing points (20). A DO loop tests for a legal grid square designator from the DATA table LT until a grid square designator is found which equals the grid square designator L. When this equality if found, control is transferred to

statement 1115 where the number of squares (N) is determined to be the loop counter minus one. A normal RETURN is executed and control transferred to the calling subroutine. If no match is found for the one letter grid square designator, control is transferred to the calling subroutine via error return RETURN A1.

Upon entry at ENTRY CNL a test is made on the square counter N. If the square counter (N) is less than zero a RETURN A1 is taken. If N is not less than zero, I, the number of legal grid square designators, is set to 23 if easting is being determined, and set to 19 if northing is being determined. Next the dispatching number, N1, is determined. Then the letter for the desired grid square designator LL is determined from the dispatch table LT. A normal exit is taken and control transferred to the calling subroutine.


SUBROUTINE CLN(LL,N,O),RETURN(A1)
L=L7(1)
CALL RVC(1,1,LL,1,L)
I=24
IF(O .EQ. ON)
F
T
I=20
DO 1110 N=1,I
IF(L .EQ. LT(N))
F
T
GO TO 1115
1110 CONTINUE
RETURN A1
1115 N=N-1
RETURN
ENTRY CLN
2
Move first character from word LL to L
Set squares index
Are northing squares being counted?
Reset square index
DO over grid square designators
Is this letter dispatchable?
Calculate square count


Is the grid square count less than zero?
Set square index for easting
Are northing squares being counted
Reset square index to northing
Find dispatch index
Set one letter grid designator
2
IF(N .LT. 0)
F
T
RETURN A1
I=23
IF(D .EQ. ON)
F
T
I=19
N1=MOD(N,I+1)+1
LL=LT(N1)
RETURN
END

1. NAME: CØØRT (other entry points: CVLN, CVNB, CITA, CITN, SETØRG)

2. TYPE OF PROGRAM: SUBROUTINE, MATHEMATICAL

3. LANGUAGE: FØRTRAN EXTENDED

4. PURPOSE: To calculate the data required for definition of a coordinate system origin and facilitate the conversion from relative easting and northing to UTM and UTM to relative easting and northing for this system.

5. ASSUMPTIONS AND LIMITATIONS: The data required for conversion of UTM to relative easting and northing coordinate must be entered via the argument list. All other conversions must have the appropriate data in common.

6. ERROR RETURNS: A RETURN A1 will be executed if either of the following errors are detected:

   A. An illegal 2 letter UTM grid square designator has been found.

   B. Point being converted from UTM to relative easting and northing lies west or south of the coordinate system origin.

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   entry point CALL CVNB (X, Y, L)

   entry point CALL CITA (X,Y, L), RETURNS (A1)

   entry point CALL CITN (X, Y, L), RETURNS (A1)

   entry point CALL SETØRG (X, Y, L), RETURNS (A1)

   X - Relative easting of the point under consideration.

   Y - Relative northing of the point under consideration

   L - Two letter UTM grid square designator.

   A1 - Control is transferred to this statement number if an error is detected in CØØRT.

8. PROGRAMS CALLING THIS PROGRAM:

   Entry point CVNB: MAIN

   Entry point CITA: MAIN, ØNEPT

   Entry point CITN: MAIN, ØNEPT

   Entry point SETØRG: MAIN

9. COMMON VARIABLES USED:

   A. ZZCV1 - IØRG, JØRG, LTØRG, XØRG, YØRG

   B. ZZCV2 - LTRS, NBRS, NBRBCD

10. COMMON VARIABLES TO BE SET:

   A. ZZCV1 - IØRG, JØRG, LTØRG, XØRG, YØRG

   B. ZZCV2 - LTRS, NBRS

11. COMMON VARIABLES CHANGED:

   A. ZZCV1 - IØRG, JØRG, LTØRG, XØRG, YØRG

   B. ZZCV2 - LTRS, NBRS, NBRBCD

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

   A. SUBROUTINES

      (1) CALL CLN (LL, N, D), RETURNS (AI) - Calculates the number of 100 km grid squares between the reference point AA and and specified grid square designator.

      LL - Two letter UTM grid square designator.
      N - Integer, number of 100 km grid squares between the specified point and the reference point (AA).

      D - Indicates whether the squares to be counted are easting (E) or northing (N).

A1 - Exit is taken when an illegal grid square designator is detected in CLN.

(2) CALL CNL (LL, N, D), RETURNS (A1) (entry point to CLN) - Determines the corresponding UTM grid square designator a specified number of 100 km grid squares from the reference point AA.

LL - One letter grid square designator corresponding to either the northing or easting direction from the reference location AA.

N - Integer, number of 100 km grid squares between the specified point and the reference point (AA).

D - Indicates whether the squares between the reference point and calculated grid square designator correspond to an easting or northing direction.

A1 - Exit is taken when an illegal grid square designator is detected in CNL.

(3) CALL EBCD (NBRS, IDUM, NBRBUF) - Converts the 8 digit UTM grid coordinate into an alphanumeric code suitable for printing.

NBRS - 8 digit UTM grid coordinate.

IDUM - Dummy variables not used.

NBRBUF - Array of dimension 3, NBRBUF (2) containing the coordinate corresponding to easting and NBRBUF (2) containing the coordinate corresponding to northing.

(4) CALL MVC (ICNT, INPTR, ZINADD, IØPTR, ZIØADD) - Move characters from the word ZINADD into the word ZIØADD.

ICNT - Integer, number of characters to be moved from ZINADD to ZIØADD.

INPTR - Integer, index of the first character to be moved from ZINADD.

ZINADD - Character, 2 word array (20 character) from which characters will be moved.

IØPTR - Integer, index of the location in ZIØADD where the first character will be moved.

ZIØADD - Character, 2 word array (20 character) from which the characters will be moved.

13. NOTES ON METHODOLOGY: Entry to CØØRT via entry point SETØRG calculates the displacement of the origin (east and north) from the UTM reference point AA. Upon entry to SETØRG, the two letter UTM grid square designator of the origin (L) is stored in the origin data table ZZCV1 and the UTM data table ZZCV2, and logical ZHØME is set to TRUE to insure origin setup. At statement 1151, CLN is called to find how many 100 km grid squares east of the reference point AA the coordinate system origin is located. The displacement of the origin in meters east of AA is then calculated and stored in integer format and real format in local variables II and XI, respectively. MVC and CLN are used to find how many squares north of the reference point the coordinate system origin is located. MVC moves the second letter of the UTM grid square designator into the first position. CLN finds the number of squares north of the reference point the origin is located. The displacement of the origin in meters north of the reference point is then calculated and stored in integer format and real format in local variables JI and YI, respectively. If ZHØME is FALSE, the displacement of the origin is stored in the origin data table and control is returned to the calling program.

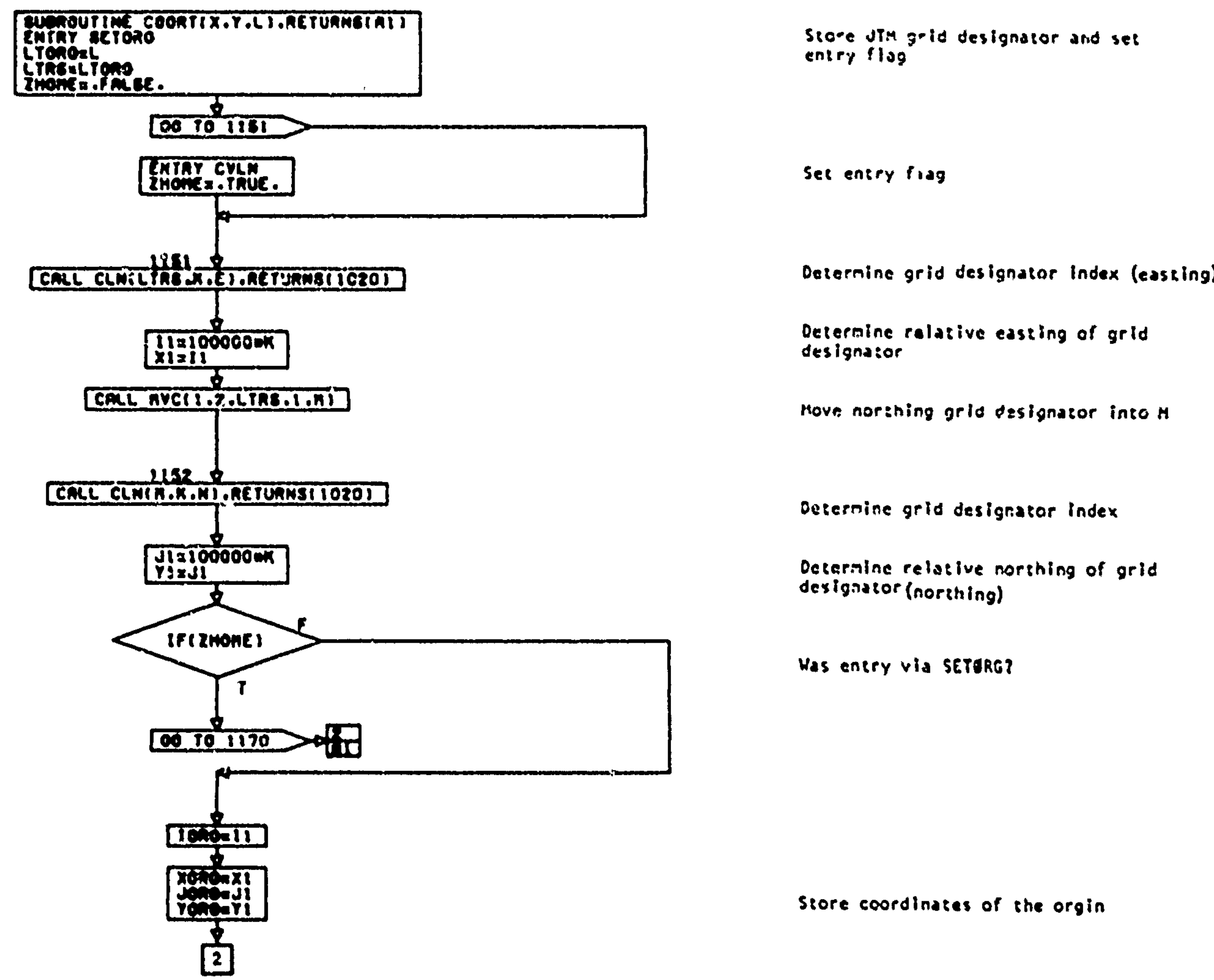
If entry to CØØRT is via entry CVLN the displacement of the 2 letter grid square designator LTRS from the reference point AA is determined. Entry CVLN sets logical ZHØME to TRUE and uses the same code as entry SETØRG to determine the east and north displacements from AA. These displacements (east and north) are stored in local variables in integer format and real format and control is returned to the calling program.

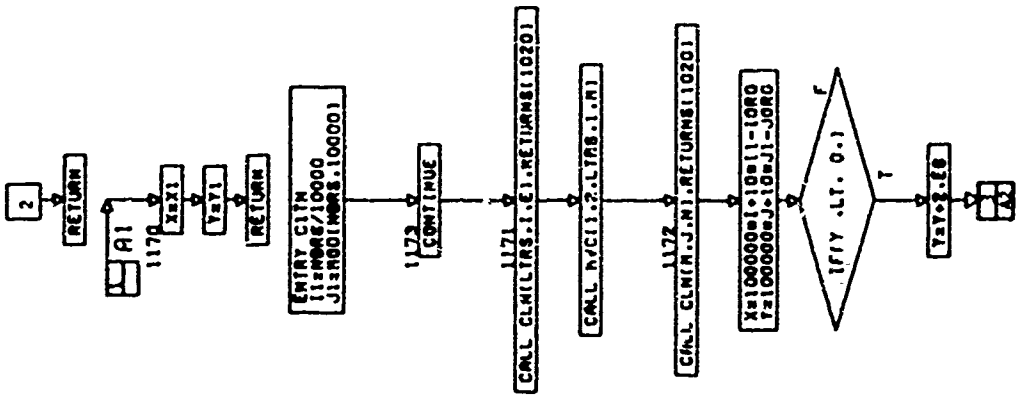
Entry via CITN converts UTM coordinates to relative easting and northing for the coordinate system established via entry SETØRG. CITN first calculates the easting and northing due to the 8 digit UTM grid coordinates (NBRS). The call to CLN at statement 1171 determines how many grid squares east of the reference point AA the grid square LTRS is located. A call to MVC and CLN finds how many grid squares north of the reference point AA the grid square LTRS is located. The relative easting (X) and northing (Y) of the point of interest is then determined by addition of these components. If the point lies west of the coordinate system origin the error exit RETURN A1 is taken. Otherwise a normal return is taken to the calling subroutine.

Entry CITA converts the relative easting and northing of a prespecified point to UTM coordinates. Upon entry via CITA, LTRS is set to a field of blanks, I is set to the number of 10 meter intervals between the specified point and the reference point measured along the east axis, and k is found to be the number of 100 km squares between the reference point and the specified point measured along the east axis. At statement 1111, CNL is called to find the easting letter for the grid square designator. Then J is set to the number of 10 meter intervals between the specified point and the reference point measured along the north axis and K is found to be the number of 100 km squares between the reference point and the specified point measured along the north axis. CNL is called at statement 1112 to find the northing letter corresponding

to this square. MVC is then called to mask together the easting letter (LTRS) and the northing letters (M) to form the UTM grid square designator (LTRS) for this point. The 8 digit UTM grid coordinate for this point is then found and stored in NBRS. EBCD is then called to convert the 8 digit grid coordinate into an alphanumeric code suitable for printing. EBCD expands the 8 digit grid coordinates into two 4 digit variables NBRBUF (2) and NBRBUF (3). These grid coordinates are then stored in the UTM data table in locations NBRBCD (1) and NBRBCD (2), respectively, and control returned to the calling program. If an error is detected during an attempt to convert to UTM coordinates an error RETURN A1 will be executed.

Entry CVNB converts the 8 digit grid coordinate NBRS to an alphanumeric code suitable for printing using the same code as was used by entry CITA.


SUBROUTINE COORT(X,Y,L),RETURNS(A1)
ENTRY SETORO
LTORO=L
LTRS=LTORO
ZHOME=.FALSE.
Store UTM grid designator and set entry flag
GO TO 1151
ENTRY CVLN
ZHOME=.TRUE.
Set entry flag
1151
CALL CLN(LTRS,K,E),RETURNS(1020)
Determine grid designator index (easting)
I1=100000*K
X1=I1
Determine relative easting of grid designator
CALL MVC(1,2,LTRS,1,M)
Move northing grid designator into M
1152
CALL CLN(M,K,N),RETURNS(1020)
Determine grid designator index
J1=100000*K
Y1=J1
Determine relative northing of grid designator (northing)
IF(ZHOME)
F
T
Was entry via SETØRG?
GO TO 1170
IORO=I1
XORO=X1
JORO=J1
YORO=Y1
Store coordinates of the orgin
2


2
RETURN
A1
1170
Xa=X1
Ya=Y1
RETURN
ENTRY CLIN
I1=NORS/10000
J1=MOD(NORS,100001)
1173
CONTINUE
1171
CALL CLN(LTRS,I,E),RETURNS(10201)
CALL M/C(1),2,LTRS,1,N)
1172
CALL CLN(N,J,N),RETURNS(10201)
X=100000*I+10*(I-10RG
Y=100000*J+10*(J1-JORG
IF(Y .LT. 0.)
F
T
Ya=Y+2.E6

Is easting positive?

Set grid designator to blanks

Determine easting grid square index

Place one letter easting grid designator in LTR8

Determine northing grid square index

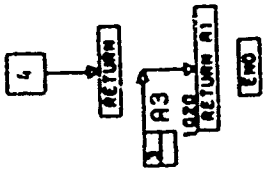
Place one letter northing grid designator

Place northing grid designator in LTR8

Calculate 5 digit UTM grid coordinate

Code NBRS for printing

Store 5 digit UTM grid coordinate


4
RETURN
A3
1020
RETURN A1
END

1. NAME: CVNBM

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE USED: FORTRAN EXTENDED

4. PURPOSE: Control the conversion of a pair of UTM grid coordinates to alphanumeric code suitable for printing.

5. ASSUMPTIONS AND LIMITATIONS: The arguments N1 and N2 are both 8 digit grid coordinates in integer format.

6. ERROR RETURNS: None

7. LINKAGE STATEMENTS AND DESCRIPTION OF ARGUMENTS:

   CALL CVNBM (N1, N2)

   N1 - Integer, 8 digit UTM grid coordinate of the lower left corner of an area map

   N2 - Integer, 8 digit UTM grid coordinate of the upper right corner of an area map.

8. PROGRAMS CALLING THIS PROGRAM: CML, DMHD

9. COMMON VARIABLES USED:

   A. ZZCV3 - NCL, NCU

10. COMMON VARIABLES TO BE SET: None

11. COMMON VARIABLES CHANGED:

    A. ZZCV3 - NCL, NCU

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

    A. SUBROUTINES

       (1) CALL EBDC (NBRS, IDUM, NBRBUF) Converts the 8 digit integer UTM grid coordinate into a two 4 digit UTM grid coordinate representation.

NBRS - Integer, 8 digit UTM grid coordinate of either the lower left corner or upper right corner of an area map.

IDUM - Integer, dummy variable not used.

NBRBUF - Integer, array of dimension 3, NBRBUF (2) being the 4 digit UTM easting coordinate and NBRBUF (3) being the 4 digit UTM northing coordinate.

13. NOTES ON METHODOLOGY: Upon entry to CVNBM a call is made to EBCD. This call converts the 8 digit integer UTM grid coordinate to a two 4 digit code. The arguments calculated via this call are: IA (2) integer, 4 digit UTM easting grid coordinate for the lower left corner; IA (3), integer 4 digit UTM northing grid coordinate for the lower left corner. Next NCL (1) is set to IA (2) and NCL (2) is set to IA (3). A call is then made to EBCD to convert the 8 digit code N2 for the upper left corner. The arguments calculated are then set to NCU (1) and NCU (2), respectively. Control is then returned to the calling routine.

SUBROUTINE CVNBM(N1,N2)
CALL EDCDI(N1,0,IA)
MCL(1)=IA(2)
MCL(2)=IA(3)
CALL EDCDI(N2,0,IW)
MCU(1)=IA(2)
MCU(2)=IA(3)
RETURN
END
Code 6 digit UTM grid coordinate for printing
Store code for UTM grid coordinate
Code 8 digit UTM grid coordinate for printing
Store code for UTM grid coordinate

1. NAME: EBCD

2. TYPE OF PROGRAM: SUBROUTINE, DATA

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Convert the 8 digit UTM grid coordinate into an alpha-numeric code suitable for printing.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL EBCD (NBRS, IDUM, NBRBUF)

   NBRS - 8 digit numerical part of the UTM coordinate

   IDUM - Dummy variable.

   NBRBUF - Integer, array of dimension 3, NBRBUF (2) containing the easting coordinate and NBRBUF (3) containing the northing coordinate.

8. PROGRAMS CALLING THIS PROGRAM: CØØRT, LABEL, MAIN

9. COMMON VARIABLES USED: None

10. COMMON VARIABLES TO BE SET: None

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None

13. NOTES ON METHODOLOGY: EBCD encodes and separates the 8 digit UTM grid coordinate into two 4 digit variables representing the easting and northing of the 8 digit grid coordinate. First, ENCODE is used to encode the 8 digit integer coordinate and store this number in display code in NBRBUF (3). NBRBUF (3) then contains the 10 character display code (2 blanks, 8 digits) representing the grid coordinate. This code is then shifted right 4 characters to drop off the last 4 characters of the word, and then

shifted right 6 characters and stored in NBRBUF (2). This procedure stores a 10 digit display code (4 characters, 6 blanks) representing the easting grid coordinate in NBRBUF (2). The code in NBRBUF (3) is then shifted left 6 characters and stored in NBRBUF (3). This procedure stores a 10 digit display code (4 character, 6 blanks) representing the northing grid coordinate in NBRBUF (3). Control then returns to the calling program.

Encode UTM grid coordinates
Store UTM grid coordinates

```
SUBROUTINE  EBCDI(HORS,IOUR,HORBUF)
ENCODE(10,100,HORBUF(31))  HORS
HORBUF(2)=SHIFT(SHIFT(HORBUF(31),-24),30)
HORBUF(3)=SHIFT(HORBUF(31),30)
```

RETURN

END

1. NAME: ELEV

2. TYPE OF PROGRAM: SUBROUTINE, MATHEMATICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Perform a bilinear interpolation to determine terrain elevation.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: A RETURN A1 will be executed if ØNEPT is unable to access terrain data for the point under consideration.

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL ELEV (XS,YS,Z), RETURNS (A1)

   XS - Relative easting of the point under consideration.

   YS - Relative northing of the point under consideration.

   Z - MSL altitude of the point under consideration.

   A1 - Exit taken if the condition listed under 6 (above) occurs.

8. PROGRAMS CALLING THIS PROGRAM: MAIN

9. COMMON VARIABLES USED:

   A. ZZCV2 - LTRS, NBRS

10. COMMON VARIABLES TO BE SET:

    A. ZZCV2 - LTRS, NBRS

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

    A. SUBROUTINES

       (1) CALL ØNEPT(XS,YS,H), RETURNS(A1) - Calculate the terrain elevation at the specified grid point.

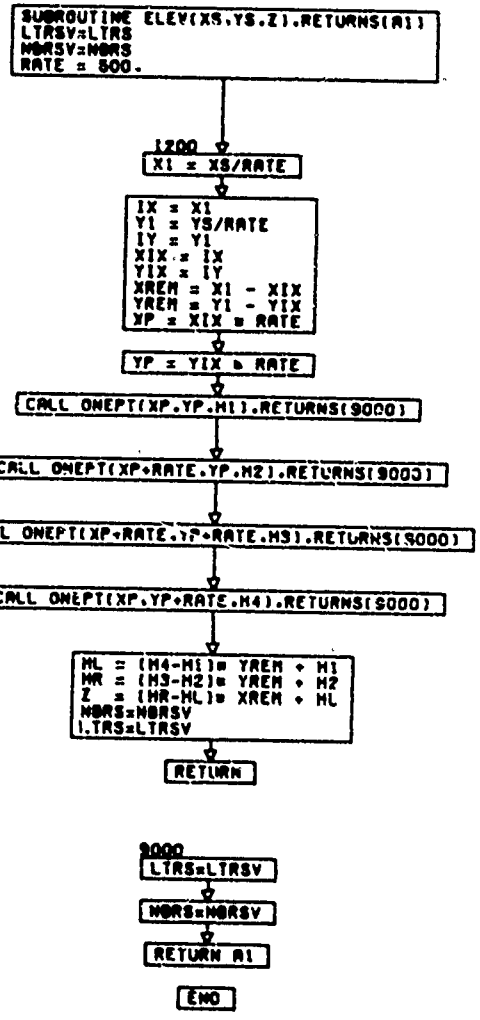
           XS - Relative easting of the grid point under consideration.

YS - Relative northing of the grid point under consideration.

H - MSL altitude of the terrain at the grid point under consideration.

A1 - Control is transferred to this statement number if an error is detected in ØNEPT.

13. NOTES ON METHODOLOGY: Upon entry to ELEV, the UTM coordinates of the point under consideration are stored in local variables (LTRSV, NBRSV) and the lateral data rate is set to 500 meters. The easting and northing of the closest western and southern grid lines are calculated. The easting (XREM) and northing (YREM) from these grid lines to the point of interest and the relative easting (XP) and northing (YP) of the intersection of these grid lines are calculated. ØNEPT is then called four times to calculate the MSL altitude at each of the four corners of the square formed by the intersection of the four grid lines closest to the point of interest. These altitudes are stored in local variables H1, H2, H3, and H4. The terrain elevation (z) is then calculated by bilinear interpolation and control returned to the calling program.



Store point location in UTM
Store lateral data rate

Calculate relative easting and northing of surrounding grid lines and point distances from these grid lines

Calculate ground altitude at grid intersection southwest of point

Calculate ground altitude at grid intersection southeast of point

Calculate ground altitude at grid intersection northeast of point

Calculate ground altitude at grid intersection northwest of point

Perform bilinear interpolation to calculate altitude of point
Store point location in UTM

Store point location in UTM

1. NAME: ERR
2. TYPE OF PROGRAM: SUBROUTINE, OUTPUT
3. LANGUAGE: FORTRAN EXTENDED
4. PURPOSE: Output to the printer an error message indicating the subroutine and statement number where an error occurred. Also issues CALL PDUMP to dump variables in this routine.
5. ASSUMPTIONS AND LIMITATIONS: None
6. ERROR RETURNS: None
7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL ERR (NAME, LAST, NSTAT)

   NAME - The name of the subroutine where error was detected.

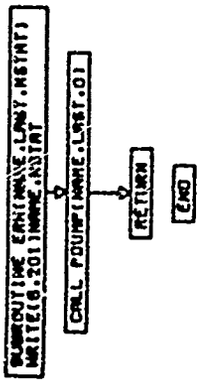
   LAST - End address of the storage area to be dumped from subroutine NAME.

   NSTAT - Statement number in subroutine NAME where error occurred.
8. PROGRAMS CALLING THIS PROGRAM:

ABSCOR	MAIN
LABEL	PMAP

9. COMMON VARIABLES USED: None
10. COMMON VARIABLES SET: None
11. COMMON VARIABLES CHANGED: None
12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None
13. NOTES ON METHODOLOGY: Subroutine ERR prints the name of the subroutine and statement number where an error occurred. Then a call is made to PDUMP for a dump of the variables used by this subroutine.


Print error message
Dump commons


SUBROUTINE ERRTRN(E,LUNIT,NSTAT)
WRITE(6,201)NAME,NSTAT
CALL FOUNFT(NAME,LAST,01)
RETURN
END

1. NAME: JTRK

2. TYPE OF PROGRAM: FUNCTION, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Find the checkpoint code from the track data array.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:
   X = JTRK (NTK,I)
   NTK - Integer, track index of the point of interest.
   I - Integer, leg index of the point of interest.

8. PROGRAMS CALLING THIS PROGRAM: None

9. COMMON VARIABLES USED: None

10. COMMON VARIABLES TO BE SET: None

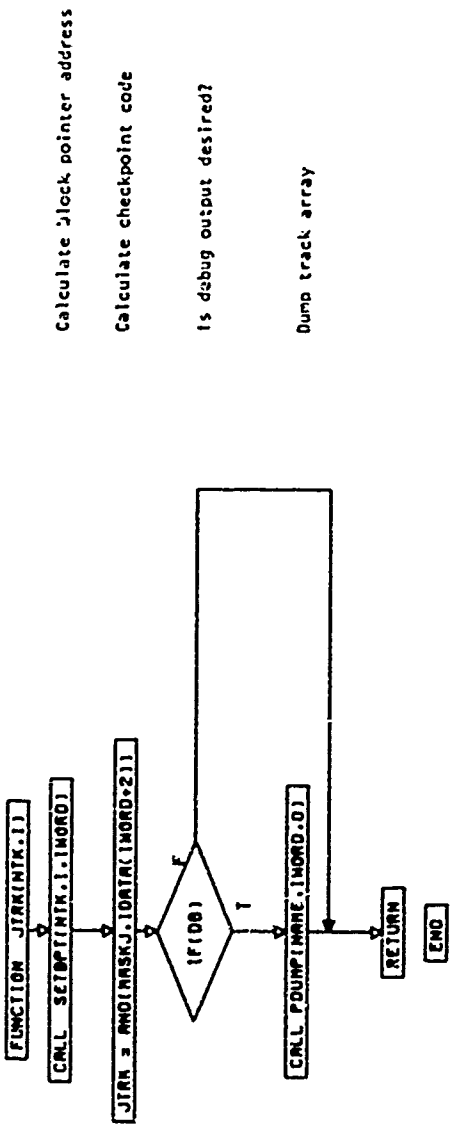
11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

    A. SUBROUTINES

       (1) CALL SETBPT (NTK, I, IWØRD)
           NTK - Index of desired track
           I - Index of desired point
           IWØRD - Block pointer address for the desired point

13. NOTES ON METHODOLOGY: Upon entry to JTRK, the block pointer word for this track point is calculated by calling SETBPT. JTRK is then set to the checkpoint code for this track point. The debug flag (DB) is then tested. If the debug flag is TRUE, PDUMP is called to dump the track data array and control returns to the calling program.


Calculate block pointer address
Calculate checkpoint code
Is debug output desired?
Dump track array
FUNCTION JTRK(NTK,I)
CALL SETUP(NTK,I,IWORD)
JTRK = AND(MASK,IDATA(IWORD+2))
IF(DB)
F
T
CALL POUMP(NAME,IWORD,0)
RETURN
END

1. NAME: LABEL

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Determine the UTM grid square letter for PMAP label.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: A call to ERR and a RETURN A1 will be executed if the following error is detected:

   A. CNL cannot find a legal grid letter corresponding to X.

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS

   CALL LABEL (M, X, XØRG, ID), RETURNS (A1).

   M - Character, contains the one letter three digit grid label for the line X.

   X - Relative easting (ID=E) or northing (ID=N) for which the UTM grid letter is desired.

   XØRG - Displacement (meters) of the origin from the UTM reference point (AA).

   ID - Character, E indicates easting of line is desired. N indicates northing of line is desired.

   A1 - Exit taken if one of the conditions listed under 6 (above) occurs.

8. PROGRAMS CALLING THIS PROGRAM: ABSCØR, MAP

9. COMMON VARIABLES USED: None

10. COMMON VARIABLES TO BE SET: None

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

A. SUBROUTINES

(1) CALL CLN (LL, N, D), RETURNS (A1) - Calculates the number of 100 km grid squares between the reference point AA and specified grid square designator.

LL - Two letter UTM grid square designator.

N - Integer, number of 100 km grid squares between the specified point and the reference point (AA).

D - Indicates whether the squares to be counted are easting (E) or northing (N).

A1 - Control is transferred to this statement when an illegal grid square designator is detected in CLN.

(2) CALL EBCD (NBRS, IDUM, NBRBUF) - Converts the 8 digit UTM grid coordinate into an alphanumeric code suitable for printing.

NBRS - 8 digit UTM grid coordinate.

IDUM - Dummy variables not used.

NBRBUF -Array of dimension 3, NBRBUF (2) containing the coordinate corresponding to easting and NBRBUF(3) containing the coordinate corresponding to northing.

(3) CALL ERR (NAME, LAST, NSTAT) - Prints an error message indicating name of subroutine and statement number where error occurred.

NAME - Name of subroutine where error was detected.

LAST - End address of the storage area to be dumped from subroutine NAME.

NSTAT - Statement number in subroutine NAME where error occurred.

(4) CALL MVC (ICNT, INPTR, ZINADD, ØIPTR, ZIØADD) - Move a character string from the word ZINADD into the word ZIØADD

ICNT - Integer, number of characters to be moved from ZINADD to ZIØADD.

INPTR - Integer, index of the first character to be moved from ZINADD.

ZINADD - Character, 2 word array (20 character) from which characters will be moved.

IØPTR - Integer, index of the location in ZIØADD where the first character will be moved.

ZIØADD - Character, 2 word array (20 characters) where the characters will be moved.

13. NOTES ON METHODOLOGY: Upon entry to LABEL, the distance from the UTM reference point (AA) to the line X is determined for either a north-south line (ID=E) for an east-west line (ID=N). The number (M) of 100 km squares between the reference and this line is then determined. At statement 1001, CNL is called to calculate M, the grid letter corresponding to M. If this operation does not result in an error, control transfers to statement 1002, where IX is determined as the number of 100 meter steps between AA and the line X plus 1000. EBCD and MVC are then called to move the three digit code corresponding to either the easting or northing grid coordinate for this line. Control then returns to the calling program. If an error is detected by CNL, ERR is called and a RETURN A1 executed.

SUBROUTINE LABEL (N,X,XORG,ID),RETURNS(A1)
IX=(X+XORG)/10.
M=IX/1000
1001
CALL CNL(M,N,ID),RETURNS(1100)
Calculate the number of 100 km squares between reference and point of interest
1002
IX=(IX+5)/10*1000
Find the one letter UTM grid designator for the square containing the point of interest
CALL EBCDI(IX,-1,ICHAR)
Calculate grid coordinate index
CALL AVC(3,2,ICHAR(3),2,N)
Code coordinate for printout
RETURN
Store grid coordinate in label
1100
CALL ERRISUBNAM,ZZZZZZ,NER)
Print error message
RETURN A1
END

1. NAME: MAIN

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Control the logical checking and plotting of FRAG1 type input data.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: A call to ERR and a RETURN A1 will be executed if any of the following errors are detected:
   A. 1100 - An illegal grid square designator has been specified for the coordinate system origin.
   B. 4015 - An 1/Ø parity error was detected in an attempt to read the second copy of the TRACK type card from file 8.
   C. 4016 - An end of file was encountered in an attempt to read the second copy of the TRACK type card from file 8.
   D. 5001 - An 1/Ø parity error was detected in an attempt to read the second copy of the PØINT type card from file 8.
   E. 5002 - An end of file was encountered in an attempt to read the second copy of the TRACK type card from file 8.
   F. 1250 - An 1/Ø parity error was detected in an attempt to read the dispatch card from file 8.
   G. 3000 - An out of sequence card or end of file was encountered in an attempt to read the second copy of the SITE type card from file 8.
   H. 3010 - An 1/Ø parity error was detected in an attempt to read the second copy of the SITE card from file 8.
   I. 3080 - The number of sites entered exceeds the maximum allowable (255).
   J. A conversion to UTM or a coding of grid coordinates has resulted in an error or track storage has been exceeded.

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL MAIN, RETURNS (A1)

   A1 - Exit taken if one of the conditions listed under 6 (above) occurs.

8. PROGRAMS CALLING THIS PROGRAM: PMAP

9. COMMON VARIABLES USED:

   A. ZZSITE - AZ1A, AZ2A, IDFU, IDFUTY, XFU, YFU, ZFU
   B. ZZPLØT - ID, X, Y
   C. ZZTRK1 - IDTHV, IDTRK, MNP
   D. ZZFASC - KGZSV, LTDA, NØDA
   E. ZZCØRD - LNBUF
   F. ZZCV2 - LTRS, NBRBCD, NBRS
   G. ZZMAPP - NLPI, SCALE

10. COMMON VARIABLES TO BE SET:

   A. ZZFASC-KGZSV, LTDA, NØDA
   B. ZZCØRD - LNBUF
   C. ZZCV2 - NBRBCD

11. COMMON VARIABLES CHANGED:

   A. ZZSITE - AX1A, AZ2A, IDFU, IDFUTY, XFU, YFU, ZFU
   B. ZZPLØT - ID, X, Y
   C. ZZTRK1 - IDTHV, IDTRK, MNP
   D. ZZCØRD - LNBUF
   E. ZZCV2 - LTRS, NBRS
   F. ZZMAPP - NLPI, SCALE

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

   A. SUBROUTINES

      (1) CALL CVNB (X,Y,L), RETURNS (A1) (entry point to CØØRT) - Converts the 8 digit UTM grid coordinates into display code for printing.

X - Dummy variable not used.
Y - Dummy variable not used.
L - Dummy variable not used.
A1 - Control is transferred to this statement number if an error is detected in CVNB.

(2) CALL CITA (X,Y,L), RETURNS (A1) (entry point to CØØRT)- Converts the relative easting and northing of the point under consideration to UTM coordinates (2 letters, 8 digits).

X - Relative easting of the point under consideration.
Y - Relative northing of the point under consideration.
L - Dummy variable not used for this entry point.
A1 - Control is transferred to this statement number if an error is detected in CITA.

(3) CALL CITN (X,Y,L), RETURNS (A1) (entry point to CØØRT)- Converts UTM coordinates of the point under consideration to relative easting and northing.

X - Relative easting of the point under consideration.
Y - Relative northing of the point under consideration.
L - Dummy variable not used.
A1 - Control is transferred to this statement number if an error is detected in CITN.

(4) CALL EBCD (NBRS, IDUM, NBRBUF) - Converts the 8 digit UTM grid coordinate into an alphanumeric code suitable for printing.

NBRS - 8 digit UTM grid coordinate.
IDUM - Dummy variables not used.
NBRBUF - Array of dimension 3, NBRBUF (2) containing the coordinate corresponding to easting and NBRBUF (2) containing the coordinate corresponding to northing.

(5) CALL ELEV (XS,YS,Z), RETURNS (A1) - Determine the terrain elevation at a specified point.

XS - Relative easting of the point under consideration.
YS - Relative northing of the point under consideration.
Z - Terrain elevation at the point under consideration.
A1 - Control is transferred to this statement number if the specified point lies off available terrain.

(6) CALL ERR (NAME, LAST, NSTAT) - Prints an error message indicating name of subroutine and statement number where error occurred.

NAME - Name of subroutine where error was detected.
LAST - End address of the storage area to be dumped from subroutine NAME.
NSTAT - Statement number in subroutine NAME where error occurred.

(7) CALL MAP (Y, X, NY, MAXNY, NCH), RETURNS (A1) - Prints a map containing the points of interest.

Y - Array, relative northing of the points of interest.
X - Array, relative easting of the points of interest.
NY - Number of points to be printed on the map.
MAXNY - Maximum number of points to be printed.
NCH - Maximum number of characters in site identifier.
A1 - Control is transferred to this statement number if an error is detected in MAP.

(8) CALL MVC (ICNT, INPTR, ZINADD, IØPTR, ZIØADD) - Move a character string from the word ZINADD into the word ZIØADD.

ICNT - Integer, number of characters to be moved from ZINADD to ZIØADD.
INPTR - Integer, index of the first character to be moved from ZINADD.

ZINADD - Character, 2 word array (20 character) from which characters will be moved.

IØPTR - Integer, index of the location in ZIØADD where the first character will be moved.

ZIØADD - Character, 2 word array (20 characters) where the character will be moved.

(9) CALL PACK (I, ISITE, ZWENG) - Prints the track site engagement data on file 9.

I - Integer, track index.

ISITE - Integer, site index.

ZWENG - Logical, site-track engageability flag.

(10) CALL SEARCH (ISITE) - Prints track identifiers for engageable site-track combinations.

ISITE- Integer, site index for site under consideration.

(11) CALL SETØRG (X,Y,L), RETURNS (A1) (entry point to CØØRT) Set coordinate system origin to grid square specified.

X - Dummy variable not used.

Y - Dummy variable not used.

L - Two letter grid square designator for the origin of the area where the game is to be played.

A1 - Control is transferred to this statement number if an error is detected in SETØRG.

(12) CALL SØRTR (X,Y,NX,MASNX,NY,MØDE)- Perform sorts on the specified arrays.

X - Array for which an ascending sort, descending sort, or maximum and minimum of the array entries will be calculated.

Y - Array whose entries will be moved in an order corresponding to the sort performed on X.

NX - Integer, number of entries from the array X to be available for sort.

MAXNX - Integer, maximum number of entries upon which a sort is to be performed.

NY - Integer, number of entries in array Y which will be moved for each corresponding move in the X array.

MØDE - Integer, mode of operation for this routine, 1 implies descending sort,2 implies ascending sort, and 3 implies maximum and minimum entries only are desired.

(13) CALL STPNT (X, Y, Z, T, J), RETURNS (E1) - Stores track point data array.

X - Relative easting of the track point under consideration.

Y - Relative northing of the track point under consideration.

Z - Altitude of the track point under consideration.

T - Time required to reach the point under consideration.

J - Checkpoint code for the point under consideration.

E1 - Control is returned to this statement number if the track point data array overflows.

(14) CALL STRK (DUM, DUM, DUM, DUM, DUM), RETURNS (E1) (entry point to STPNT) - Initializes the block pointer for storage of this tracks points.

DUM - Dummy variable not used.

E1 - control is returned to this statement number if the track point data array overflows.

(15) CALL TTØRNG (RAGIN, TIZZ, TIME, NF, NTK) - Calculates the track entry time into the specified site fire volume.

RAGIN - Range of interest for the site under consideration.
TIZZ - Earliest time to be considered for entry.
TIME - Time a specified track enters a specified fire volume.
NF - Integer, index of the site under consideration.
NTK - Integer, index of the track under corsideration.

B. FUNCTIONS

(1) X = TTRK(NTK,I) - Finds the time associated with a specified track point.
NTK - Integer, index of the desired track.
I - Integer, index of the desired point on the track.

(2) X = XTRK(NTK,I) (entry point to TTRK) - Finds the relative easting of the specified track point.
NTK - Integer, index of the desired track.
I - Integer, index of the desired point on the track.

(3) X = YTRK(NTK,I)(entry point to TTRK) - Finds the relative northing of the specified track point.
NTK - Integer, index of the desired track.
I - Integer, index of the desired point on the track.

(4) X = ZTRK(NTK,I)(entry point to TTRK) - Finds the terrain elevation of the specified track point.
NTK - Integer, index of the desired track.
I - Integer, index of the desired point on the track.

13. NOTES ON METHODOLOGY: Upon entry to subroutine MAIN the default values for the plot scale (SCALE), the logical option flags, and UTM coordinates for the origin are initialized. The mathematical origin is then calculated by a call to subroutine SETØRG. Following initialization of the default values, the first input card is read from file 5 and stored in the record buffer (IBUF). If this record is not an end-of-file marker, control transfers to statement 1150 and two copies of the above mentioned record are punched on file 8. Control then transfers to statement 1100 for processing of the next input card. When an end of file is encountered on file 5, control transfers to statement 1200 where an end of file and rewind are executed for file 8. Next the site, system, track, and error counters are initialized to zero and the logical processing flags are initialized.

Statements 1250 through 1400 read the first copy of each card type from the secondary input device, determine the card's dispatch number, and control branching for the appropriate card types. At statement 1250, the card counter (N) is incremented and the first copy of the input card is read from file 8 and stored in the buffer (IBUF). If an end of file is encountered, control transfers to statement 1975 for final data processing. Otherwise, an I/Ø parity error test is performed and if an error is detected, control transfers to statement 9010 for error processing. A DO loop is then used to find the cards dispatching index. If this is a nondispatchable card type, control transfers to statement 1300 where an error message is printed indicating an illegal card type has been found, the second copy of the nondispatchable card type is read from file 8, the error counter is incremented, and control transfers to statement 1250 where the next card type is processed. Dispatchable indices of 1, 2, 3, 4, and 5 corresponding to card types of SWIT, SITE, TRAC, PØIN, and SYST, cause a corresponding transfer of control to statements 2000, 3000, 4000, 5000, and 6000.

At statement 1975, the logical flag ZSILL is tested to determine if any SITE cards have been processed. If ZSILL is TRUE, no site cards have been processed and control transfers to statement 1985. If ZSILL is FALSE and plotting is desired, a plot of these sites will be printed and control returns to the calling program. At statement 1985 logical flag ZNOPLT is tested to determine if plotting is desired. If ZNØPLT is FALSE, plotting is desired and the appropriate track points will be plotted. If no plotting is desired, control transfers to statement 1995 where the no engageability flag is tested. If no engageability is to be played, control returns to the calling program. Otherwise, if system sites and tracks were entered control transfers to statement 8000 where the engageability test is performed. Control then transfers to the calling program.

At statement 2000 the second copy of the SWITCH card is read from the secondary input device (file 8). Variables read from the SWITCH card are: KGZ, UTM grid zone designator; KØR, coordinate system origin; NLPI, number of lines per inch; SCALE, plotting scale factor; RNMAX, maximum leg length; ANGØ, maximum turn angle; ZTER, logical terrain flag; ZPUNCH, logical site punch flag; ZNØPLT, logical no plot flag; ZNØENG, logical no engageability flag, and ZNØSEC, logical no sector flag. Following statement 2000, the cosine of the maximum turning angle is calculated, the number of lines per page (NLPP) is calculated, and the mathematical origin is determined by a call to SETØRG. At statement 2100 the logical terrain flag is tested. If terrain is not to be considered by this program (ZTER = FALSE), control transfers to statement 1250. If terrain is to be considered, the terrain directory is read from the direct access device (file 4) and control transfers to statement 1250.

At statement 1975, the logical flag ZSILL is tested to determine if any SITE cards have been processed. If ZSILL is TRUE, no site cards have been processed and control transfers to statement 1985. If ZSILL is FALSE and plotting is desired, a plot of these sites will be printed and control returns to the calling program. At statement 1985 logical flag ZNOPLT is tested to determine if plotting is desired. If ZNØPLT is FALSE, plotting is desired and the appropriate track points will be plotted. If no plotting is desired, control transfers to statement 1995 where the no engageability flag is tested. If no engageability is to be played, control returns to the calling program. Otherwise, if system sites and tracks were entered. control transfers to statement 8000 where the engageability test is performed. Control then transfers to the calling program.

At statement 2000 the second copy of the SWITCH card is read from the secondary input device (file 8). Variables read from the SWITCH card are: KGZ, UTM grid zone designator; KØR, coordinate system origin; NLPI, number of lines per inch; SCALE, plotting scale factor; RNMAX, maximum leg length; ANGØ, maximum turn angle; ZTER, logical terrain flag; ZPUNCH, logical site punch flag; ZNØPLT, logical no plot flag; ZNØENG, logical no engageability flag, and ZNØSEC, logical no sector flag. Following statement 2000, the cosine of the maximum turning angle is calculated, the number of lines per page (NLPP) is calculated, and the mathematical origin is determined by a call to SETØRG. At statement 2100 the logical terrain flag is tested. if terrain is not to be considered by this program (ZTER = FALSE), control transfers to statement 1250. If terrain is to be considered, the terrain directory is read from the direct access device (file 4) and control transfers to statement 1250.

At statement 6000, the processing of SYSTEM type data cards begins. First the system counter (NØSYS) is incremented and the second copy of the SYSTEM type card is read from the secondary input device (file 8). The system identifier (IDSYS) and the maximum range of interest (RANMAX) are read from the SYSTEM type card and stored in the system data table. Control then transfers to statement 1250.

At statement 3000, the processing of SITE type data cards begins. If this is the first SITE type card to be processed, the site counter (NFU) is initialized to zero. A test is then performed on logical flag ZSILL to determine if this SITE card is being read out of sequence. If this SITE card is being read out of sequence, control transfers to statement 9200 for error processing. If this card is not out of sequence the logical flags ZERCVT, ZSP, and ZØN are then initialized. Next, the line buffer is initialized and the site altitude is set to zero. At statement 3010, the second copy of the SITE type data card is read from file 8. Variables contained on the SITE type data card are: IFTYP, type of system; IDF, site identifier; LTRS, two letter grid square designator; NBRS, eight digit UTM grid coordinate; ZF, altitude of the site above MSL; AS1, the left edge of sector; and AZ2, the right edge of sector. If an end of file is encountered or an I/Ø parity error is detected, control transfers to statements 9200 and 9220, respectively, for error processing. If no I/Ø errors are detected, control transfers to statement 3015 where the relative easting and northing of the site locations are determined by a call to CITN. If terrain is being considered, the terrain elevation (Z) is calculated by a call to ELEV and the difference between the calculated and specified terrain altitudes is calculated. If this site location is found to be off the available terrain area, ELEV sets the logical flag ZØN to FALSE. The sites

description is then printed in the site description table along with a note describing whether the site is off the available terrain. A test is then performed and if the site altitude specified is below the terrain elevation, an appropriate error message is printed. At statement 3019 the site counter (NFU) is tested. If the number of sites is zero no redundancy test is required and control transfers to statement 3080. Otherwise, a DO loop is entered which tests the site name against the site names already entered in the site description table. If this site name is unique, control transfers to statement 3050. If a redundant site name is found, control transfers to statement 3030 where the specified and redundant site locations are tested. If the site locations are the same, a message is printed indicating that a duplicate SITE card has been found and control transfers to statement 1250 where the next card type will be processed. Following statement 3040, a logical change flag (ZCH) is set TRUE and an error message printed to indicate that a duplicate card ID has been found. At statement 3050, a DO loop is used to determine if this site location is unique. If this site location is unique, control transfers to statement 3080. Otherwise, control transfers to statement 3070 where an error message is printed indicating that two sites have been found with the same location, the change flag ZCH is set to TRUE and control transfers to statement 1250. Following statement 3080, the site description is entered in the site description table and control transfers to statement 1250.

TRACK type data cards are processed starting at statement 4000. At statement number 4000, the logical flag ZSITE is tested to determine if site cards have been previously processed. If site cards have been previously processed, the plotting flag

(ZNØPLØT) is tested to determine if plotting is desired. If no sites have been processed or no plotting is desired, control transfers to statement 4005. Otherwise, the sites are sorted and subroutine MAP is called to plot the desired site map. Following statement 4005, logical flags ZSIT and ZSILL are reset to disallow further site processing and track counter (NTK) is incremented. At statement 4015, the second copy of the track type data card is read from file 8. If an end of file or I/Ø parity error is detected, control transfers to statement 9004 or 9006, respectively, for the appropriate error processing. The variables read from the TRACK type data card and stored in the track description tables are: IDTHV, penetrator identifier; IDTRK, track identifier; and MNP, number of track points. A DO loop is then used to test for redundant track identifiers in the track description table. If a redundant track identifier is found, control transfers to statement 5040 where an error message is printed to indicate that a redundant track identifier has been found. Control then transfers to statement 4030. At statement 4030 a test is performed to determine if the number of points (MN) processed for the previous track is less than zero. If no points were processed, control transfers to statement 4100 where an error message is printed. If points were processed for the previous track, STRK is called to determine the initial storage address for the data on this track. A test is then performed on the track counter (NTK), and control is transferred to statement 4200 if this is the first track card processed. If this is not the first track to be processed, the actual point count for this track and the logical track point error flags are tested. If an error is detected in either one of these values, an error message will be printed. Otherwise, the no plot flag is tested and if plotting is desired, the track points will be sorted and a track map will be plotted. At statement 4200, track data for this track will be printed.

Following statement 4300 the logical error flags and the point counter for this track will be initialized and control then returns to statement 1250 for processing of the next card type.

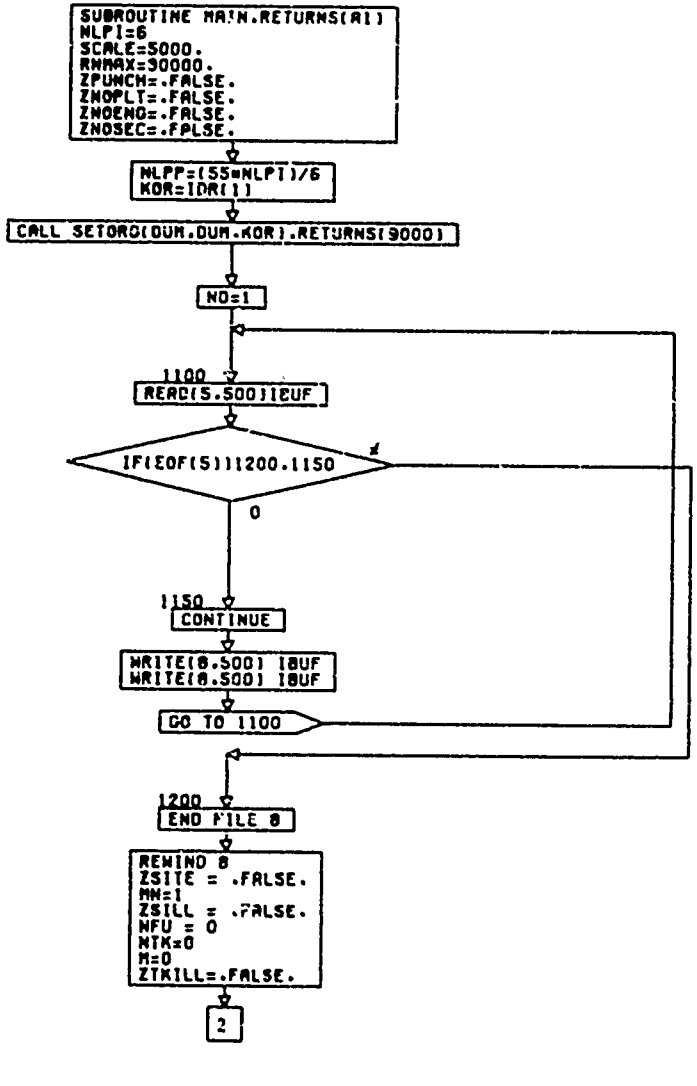
Processing of POINT type cards begins at statement 5000. Following statement 5000, the point counter (MN) is incremented and the track time (TTK) is initialized to zero. At statement 5001. the second copy of the POINT type data card is read from file 8. If an end of file or an I/Ø parity error is detected, control will transfer to statement 9008 or 9006, respectively, for error processing. The variables read from the POINT type card are: LTRS, two letter grid square designator; NBRS, eight digit UTM grid coordinate; ZTK, altitude of the track above main sea level; VEL, velocity of the vehicle approaching that point; IDCP, check point code; IDTK, track identifier; IP, sequence number of the point on the track; and IDTP, track point identifier. After the POINT type card has been successfully read from file 8, CITN is called to convert the UTM coordinates of this point to relative coordinates and CVNB is called to code the UTM grid coordinates in a form suitable for printing. The targeted track logical flags are then initialized and a test is performed to see if the track point identifier has been left blank. If the track point identifier has been left blank, this track point is not targeted for any site and control transfers to statement 5026. If a track point identifier has been specified, a loop is then entered which tests each site identifier against the track point identifier. When it is determined that a track is targeted for particular site, control transfers to statement 9310. If no targeted site is found, logical flag ZTTENG is set to FALSE and control transfers to statement 5026. At statement 9310 the square of the horizontal range between the site location and track point is calculated and tested to determine if this track point is within 25 meters of the site location. If this track point lies within

25 meters of the site location, logical flag ZWENG is set to TRUE. At statement 5026, PACK is called to punch a track/site targeting record on file 9. If this is the first point to be processed for this track, a track table heading is printed and this leg's slant range (RTK) and ground track (GRNG) are initialized. Then control transfers to statement 5010. If this is not the first point to be processed on this track, GRNG, RTK, and the time required for the vehicle to reach this point on the track (TTK) are calculated following statement 5009. If required, a skip to the top of the next page is issued and a new table heading is printed following statement 5010. At statement 5011 the track point data are printed in the track data table. If the site was not found with the specified track point identifier, an appropriate message is then printed for this track. If terrain is to be tested, a test is then performed to determine if this point lies on available terrain. If this point lies off the available terrain, an appropriate error message is printed. Statements 5013 through 5019 test this leg for four different types of errors. If this point track identifier is found to differ from the actual identifier for this track, logical flag ZTEKID is set to TRUE and an appropriate error message is printed. If the point sequence number does not correspond to the actual sequence number for this point, logical flag ZETKPT is set to TRUE and an appropriate error message is printed. If this leg's length is less than 10 kilometers, logical flag ZSP is set to TRUE and a message is printed indicating that this leg is unusually short. If RTK is greater than the maximum range specified on the SWITCH card, a message is printed indicating that an unusually long leg has been found and logical flag ZSP is set to TRUE. If the number of points processed is greater than two, a test is then performed to

determine if the turning angle defined by the previous track points is greater than the maximum expected turning angle. If the maximum expected turning angle is exceeded, ZSP is set to TRUE and an appropriate message is printed. Following statement 5020 this point is stored in the track data array and the point sequence number is stored in variable ID. Control is then returned to statement 1250. Statements 5030 through 5900 set error flags and print appropriate error messages if storage, redundant tracks, or conversion errors are detected for this point. Then control transfers to statement 1250 for processing of the next card type.

Following statement 8000, the heading for the site engageability table is printed. A major loop is then entered which determines which tracks are engageable by which sites. First the site system index (IJ) is determined. If the system type cannot be found for this site, an error message is printed and control transfers to statement 8100. Once the site system type has been determined, the site's range of interest is stored in variable RANG. A DO loop is then used to determine the tracks which intersect this fire volume. For each track, TTØRNG is called to determine the time that the vehicle flying this track will enter the fire volume. If this time is negative, the track does not enter the fire volume and control transfers to statement 8020. Otherwise, the intersection counter and intersection flags are set and the track ID is entered in the site engageability table. An appropriate engageability message is then printed for the site. Following statement 8100, a DO loop is entered which prints a table containing the identifiers for all the nonengageable tracks. If all the tracks are nonengageable by all the sites, an appropriate message is printed following statement 8200. A major DO loop is then entered which determines which tracks are targeted for which sites and

prints a table of targeted sites. First the logical flags ZTENG is set TRUE, if the track has a point which comes within 25 meters of the site location. Next a test is performed and if the track point lies within the site's range of interest, the track identifier is stored in the table ITRKSW. A message is then printed indicating which tracks are targeted for each site and control is returned to the calling program.



Initialize the number of lines per inch
Initialize scale factor

Set default punch, plot, engagement, and sector flags

Calculate the number of lines per page
Set origin in UTM
Calculate mathematical origin

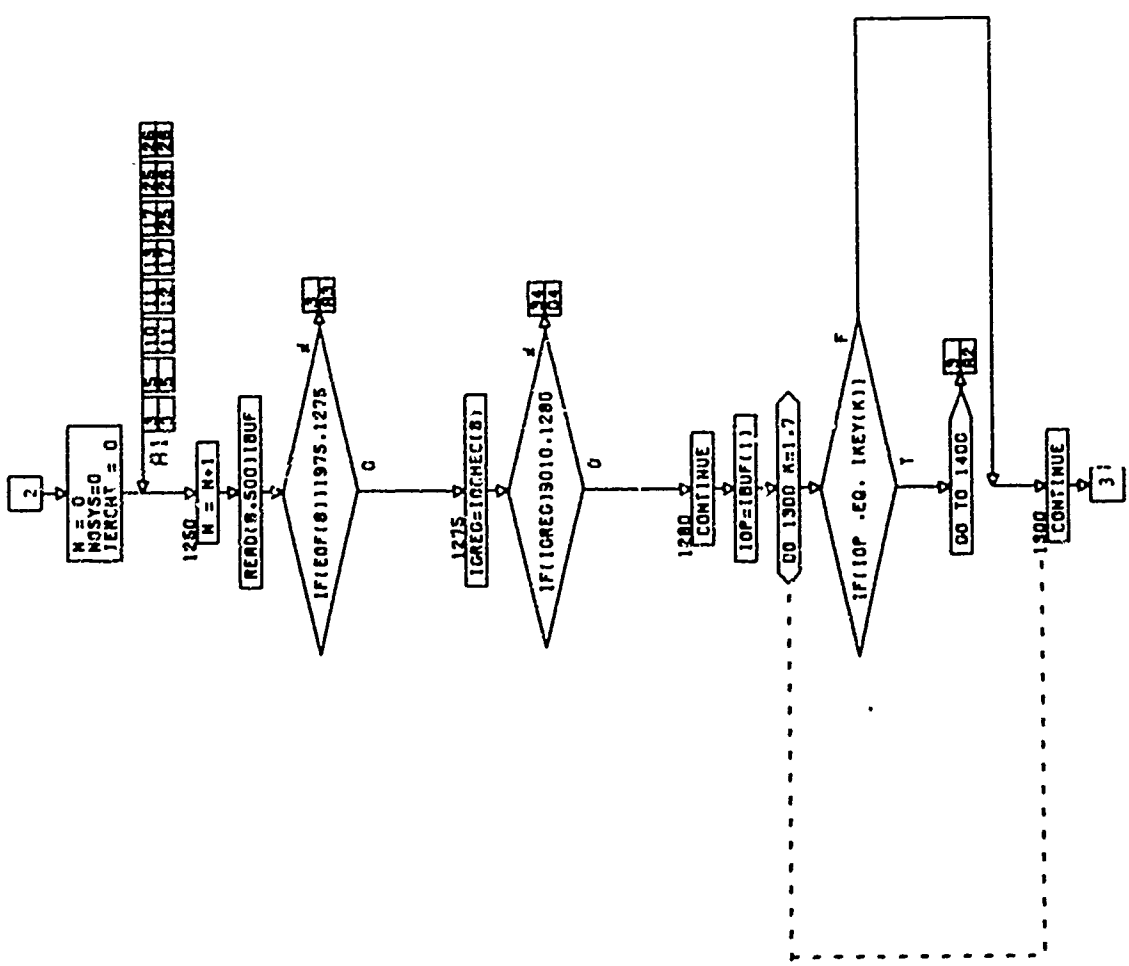
Read a card

Is EOF index zero?

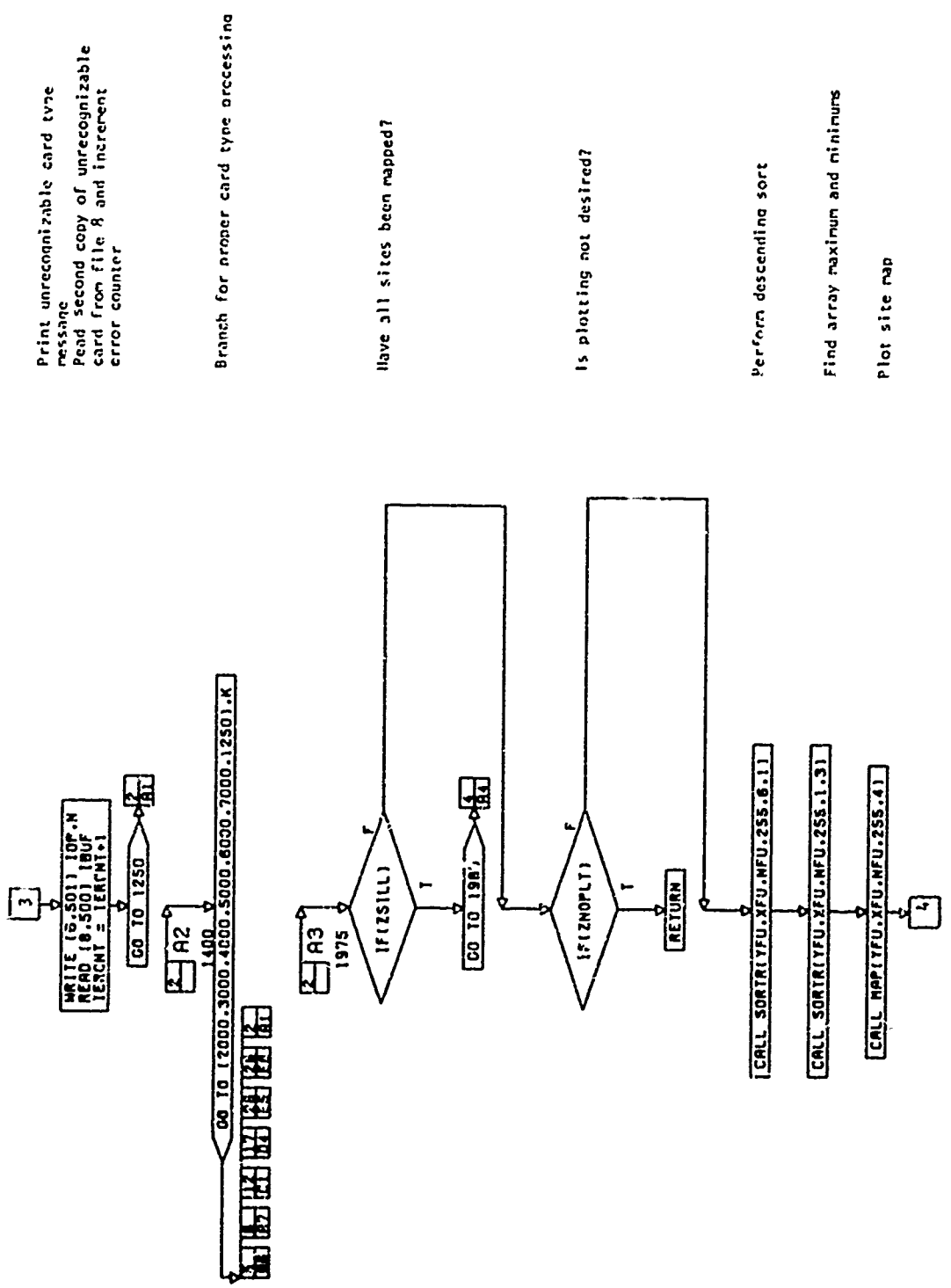
Punch two copies of this card on file 8

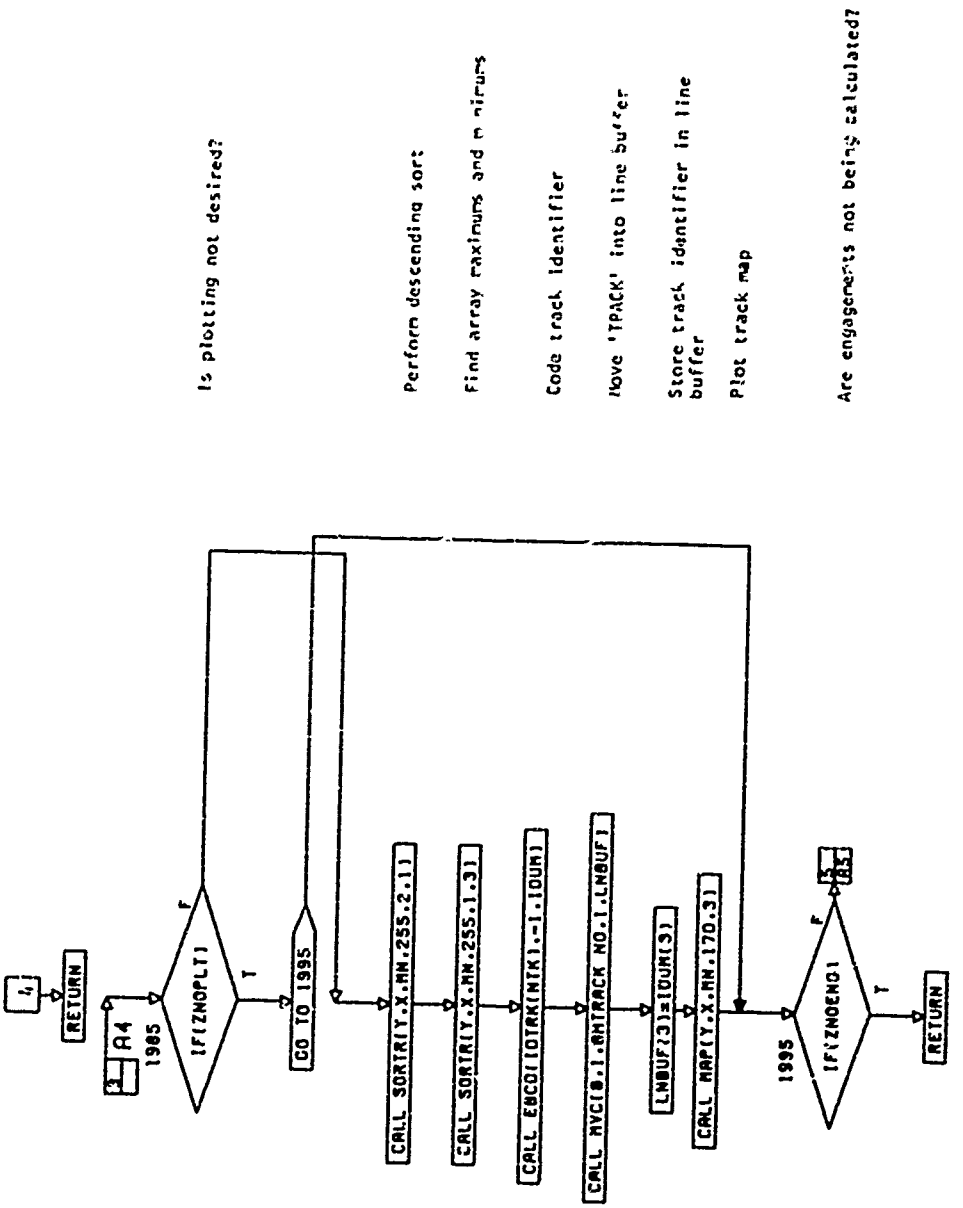
End file 8

Initialize flags and counters

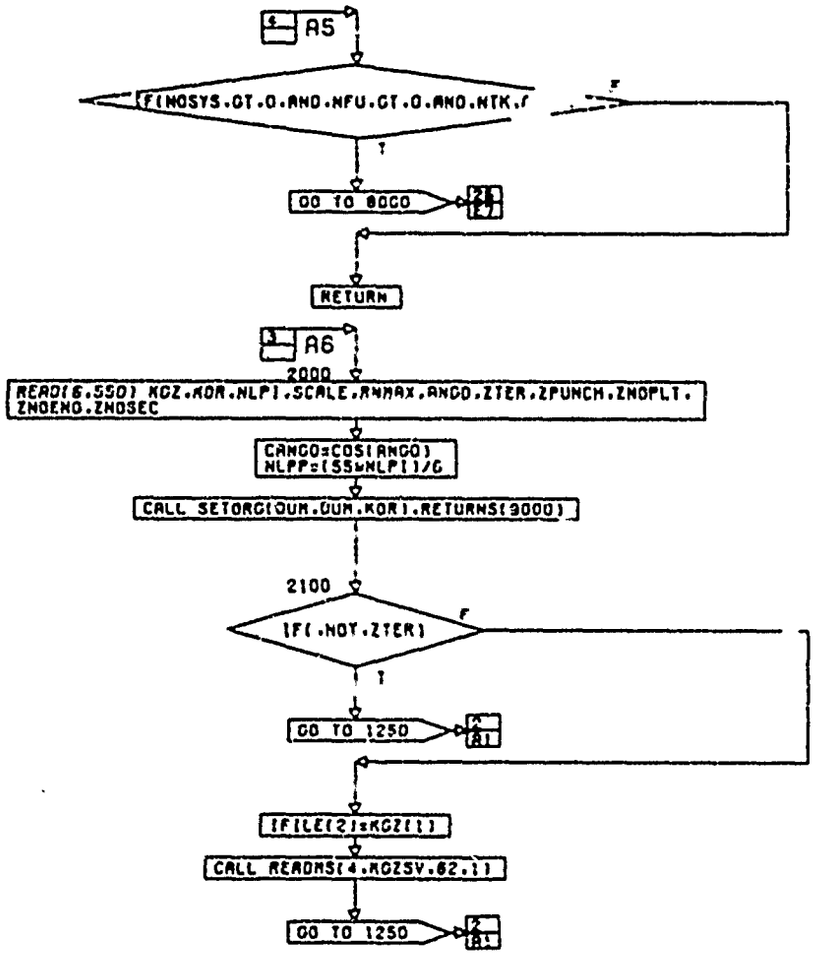

Initialize systems counter
Initialize error counter
Increment number of cards types read
Read a card from file 8
Is EOF index zero?
Set I/O parity error index
I/O parity error index zero?
Store card type
DO over dispatchable card types
Is this card dispatchable?
2
N = 0
NOSYS=0
IERCNT = 0
1250
N = N+1
READ(8,500)IBUF
IF(EOF(8))1975,1275
1275
IGREG=IORHEC(8)
IF(IGREG)3010,1280
1280
CONTINUE
IOP=IBUF(1)
DO 1300 K=1,7
IF(IOP .EQ. IKEY(K))
GO TO 1400
1300
CONTINUE
3!

P.4.MAIN-2


Print unrecognizable card type message
Read second copy of unrecognizable card from file 8 and increment error counter
Branch for proper card type accessing
Have all sites been mapped?
Is plotting not desired?
Perform descending sort
Find array maximum and minimums
Plot site map
3
WRITE (6,501) IOP,N
READ (8,500) IBUF
IERCNT = IERCNT+1
GO TO 1250
2 A2
1400
GO TO (2000,3000,4000,5000,6000,7000,12501),K
2 A3
1975
IF(ISILL)
F
T
GO TO 198
IF(ZNOPLT)
F
T
RETURN
CALL SORTR(YFU,XFU,NFU,255,6,1)
CALL SORTR(YFU,XFU,NFU,255,1,3)
CALL MAP(YFU,XFU,NFU,255,4)
4


Is plotting not desired?
Perform descending sort
Find array maximums and minimums
Code track identifier
Move 'TRACK' into line buffer
Store track identifier in line buffer
Plot track map
Are engagements not being calculated?
4
RETURN
A4
1985
IF(ZNOPLT)
F
T
GO TO 1995
CALL SORTRIY,X,MN,255,2,1)
CALL SORTRIY,X,MN,255,1,3)
CALL EBCDI(OTRK(NTK),-1,IDUM)
CALL MVC(8,1,BNTRACK NO.1,LNBUF)
LNBUF(3)=IDUM(3)
CALL MAP(Y,X,MN,170,3)
1995
IF(ZNOEND)
F
T
RETURN

P.4.MAIN-4



Have systems, sites, and tracks all been processed?

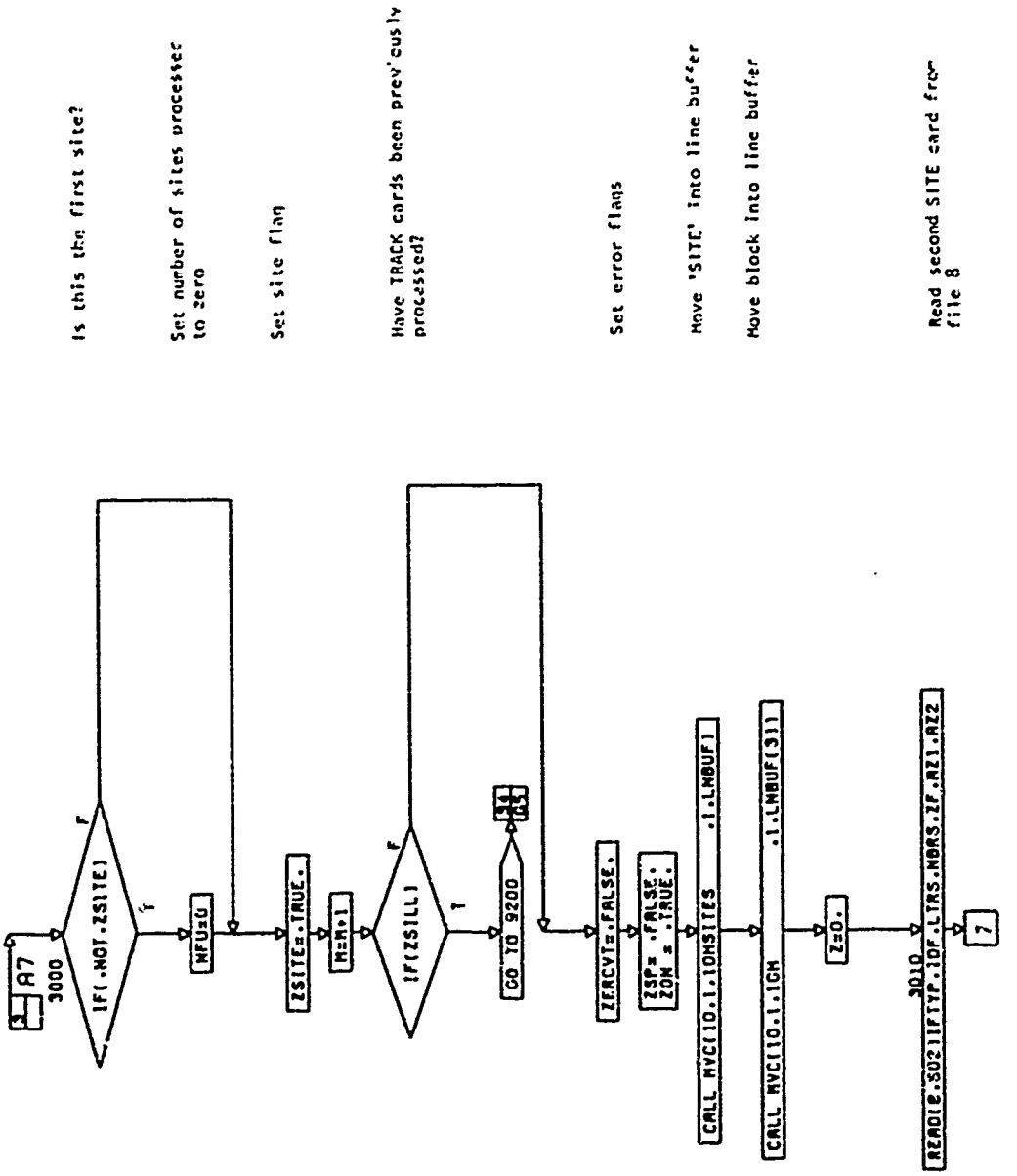
Read second SWITCH card from file 5

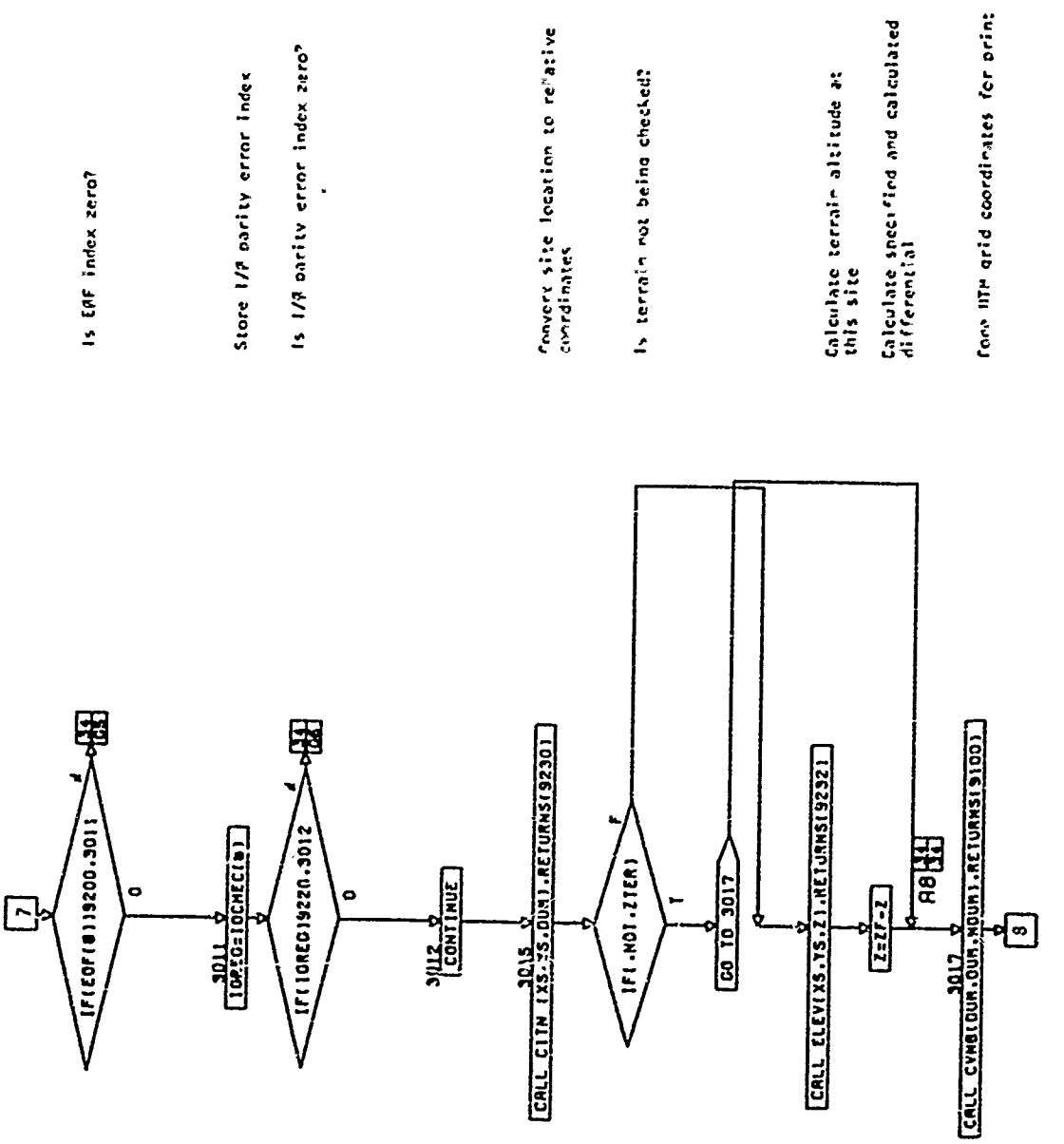
Calculate cosine of maximum turning angle and the number of lines per page
Calculate mathematical origin

Is terrain not being checked?

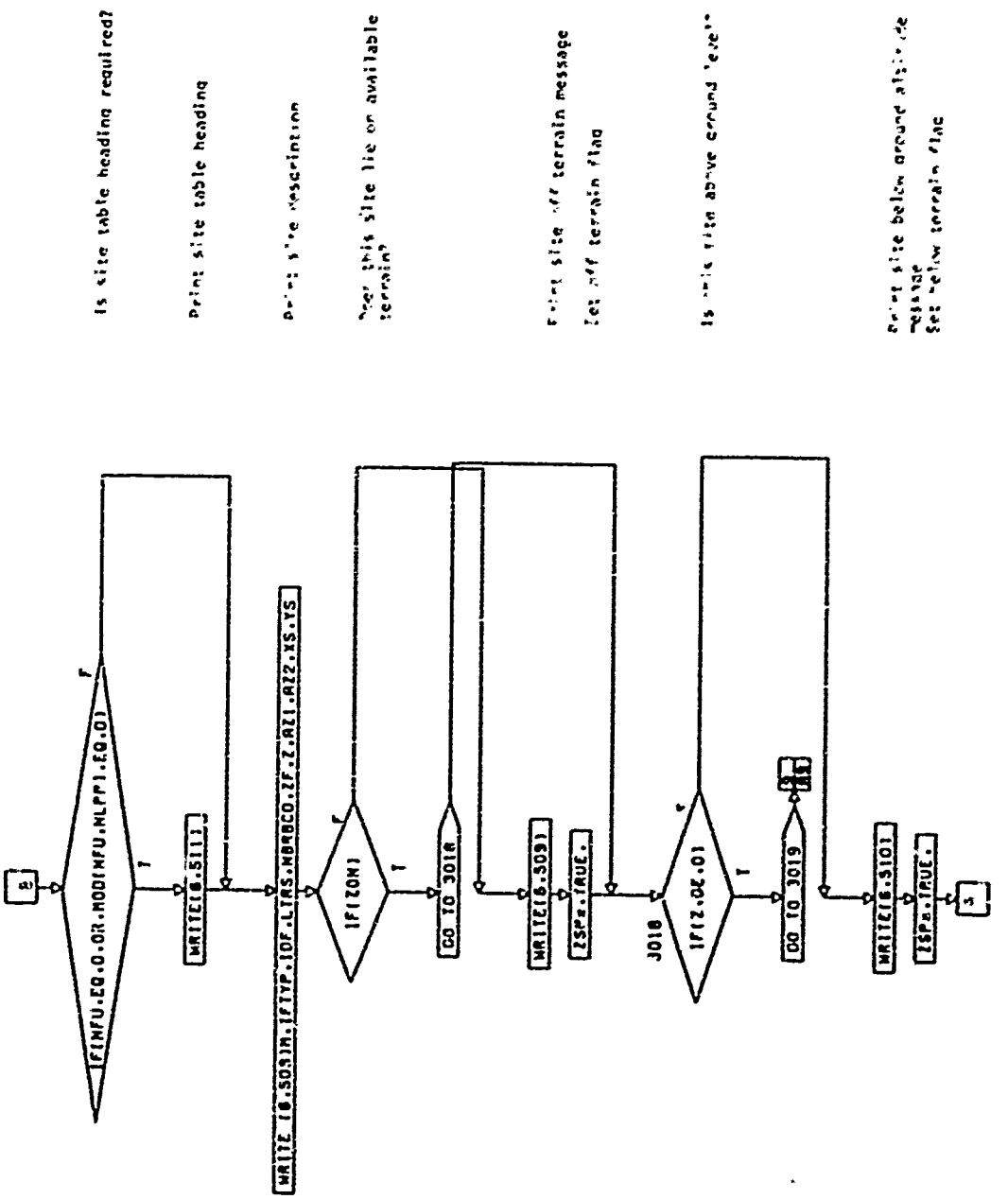
Store s location in UTM

Read te. in directory

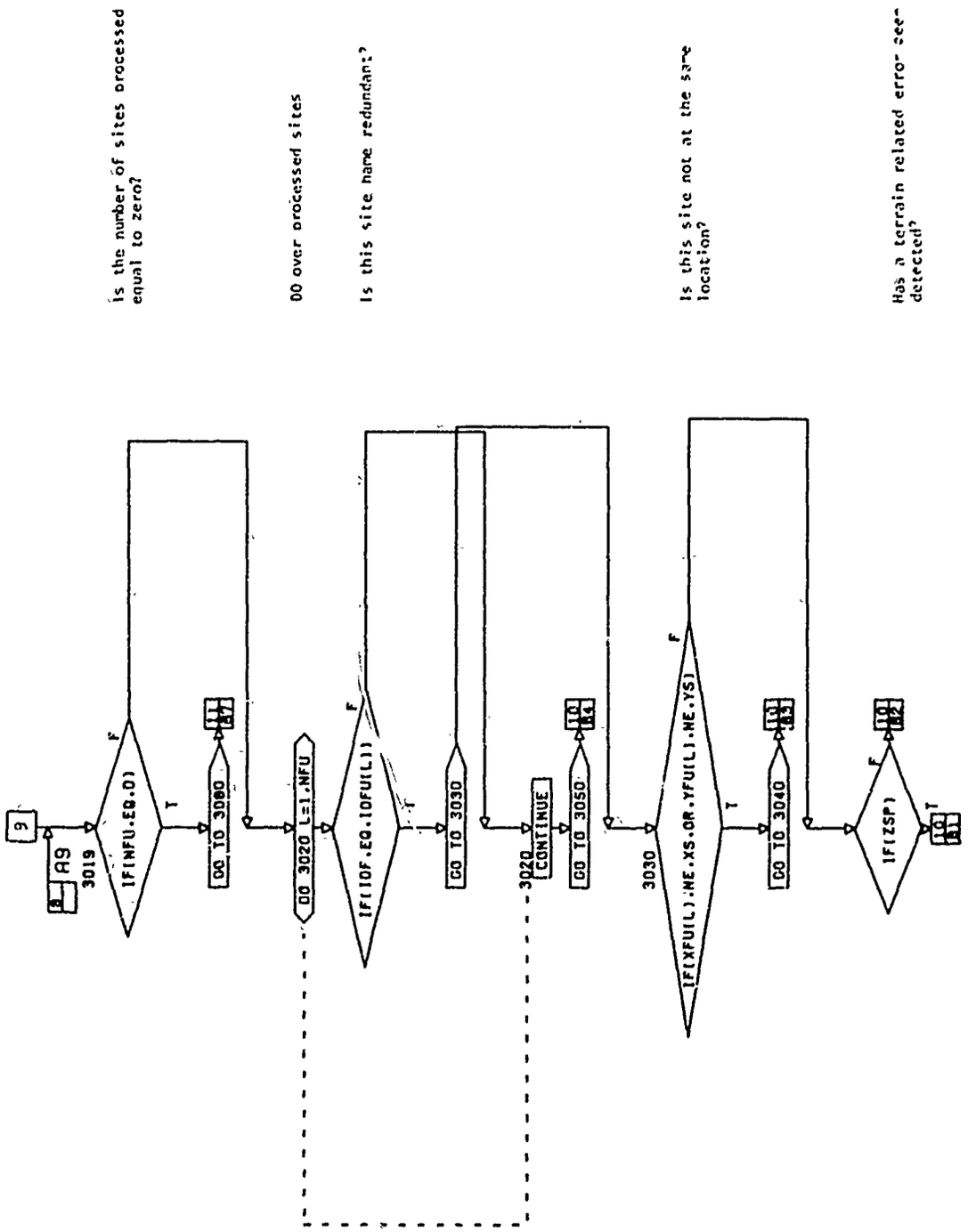

A7
3000
IF(.NOT.ZSITE)
F
T
NFU=0
Is this the first site?
Set number of sites processed to zero
ZSITE=.TRUE.
Set site flag
M=M+1
IF(ZSILL)
F
T
GO TO 9200
Have TRACK cards been previously processed?
ZERCVT=.FALSE.
Set error flags
ZSP= .FALSE.
ZON = .TRUE.
Move 'SITE' into line buffer
CALL MVC(10,1,10HSITES ,1,LNBUF)
Move block into line buffer
CALL MVC(10,1,1CH ,1,LNBUF(31))
Z=0.
READ(8,5021)FTYP,IOF,LTRS,NDAS,IF,AZ1,AZ2
Read second SITE card from file 8
3010
7

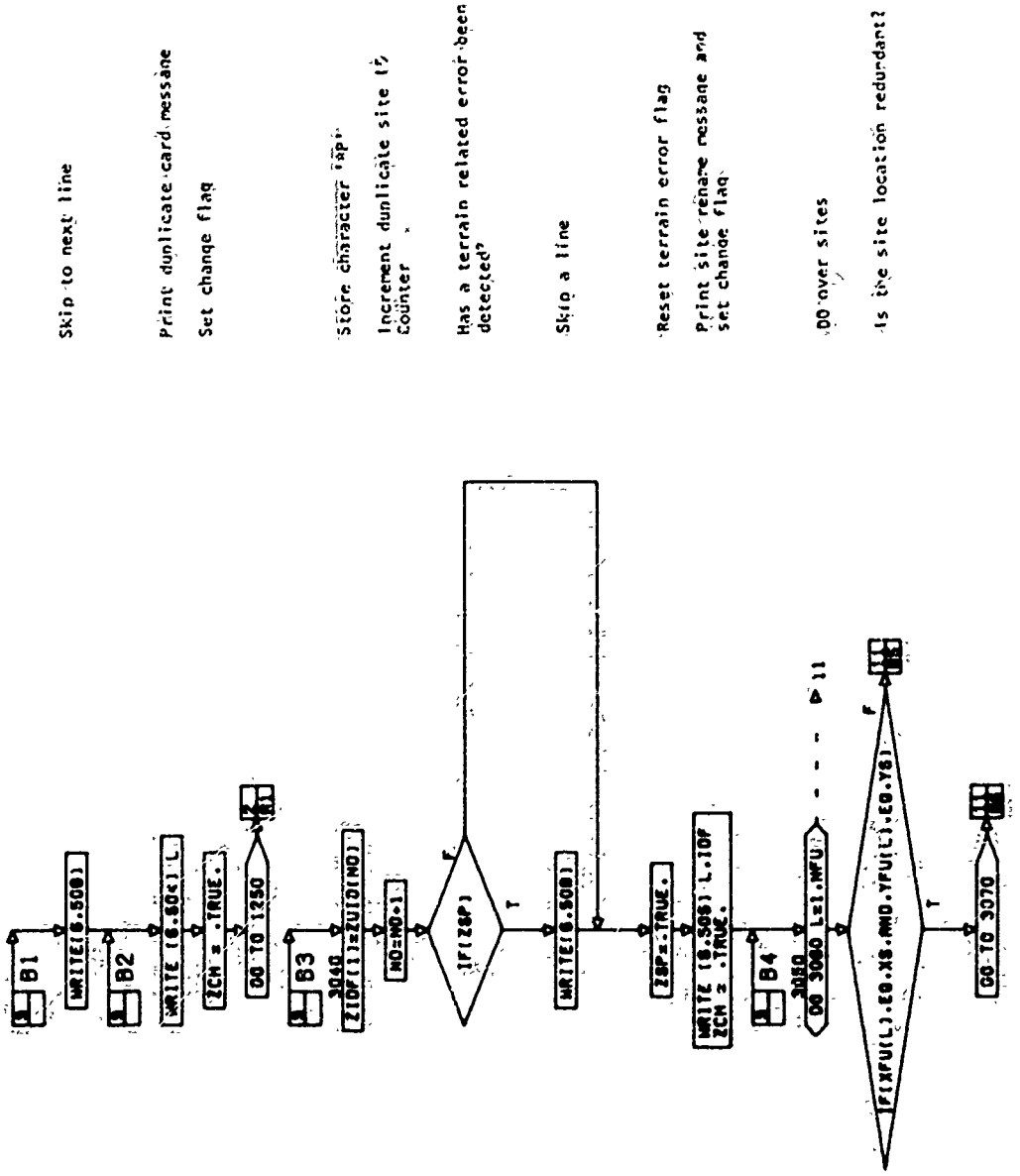

Is EOF index zero?
Store I/O parity error index
Is I/O parity error index zero?
Convert site location to relative coordinates
Is terrain not being checked?
Calculate terrain altitude at this site
Calculate specified and calculated differential
Form UTM grid coordinates for print
7
IF(EOF(8)).EQ.19200.3011
0
3011
IOR=IOCHEC(8)
IF(IOR.EQ.19220.3012
0
3012
CONTINUE
CALL CITN (XS,YS,DUM),RETURNS(92301)
IF(.NOT.ZTER)
F
T
GO TO 3017
CALL ELEV(XS,YS,Z),RETURNS(92321)
Z=ZF-Z
A8
3017
CALL CVNB(DUM,DUM,DUM),RETURNS(91001)
8

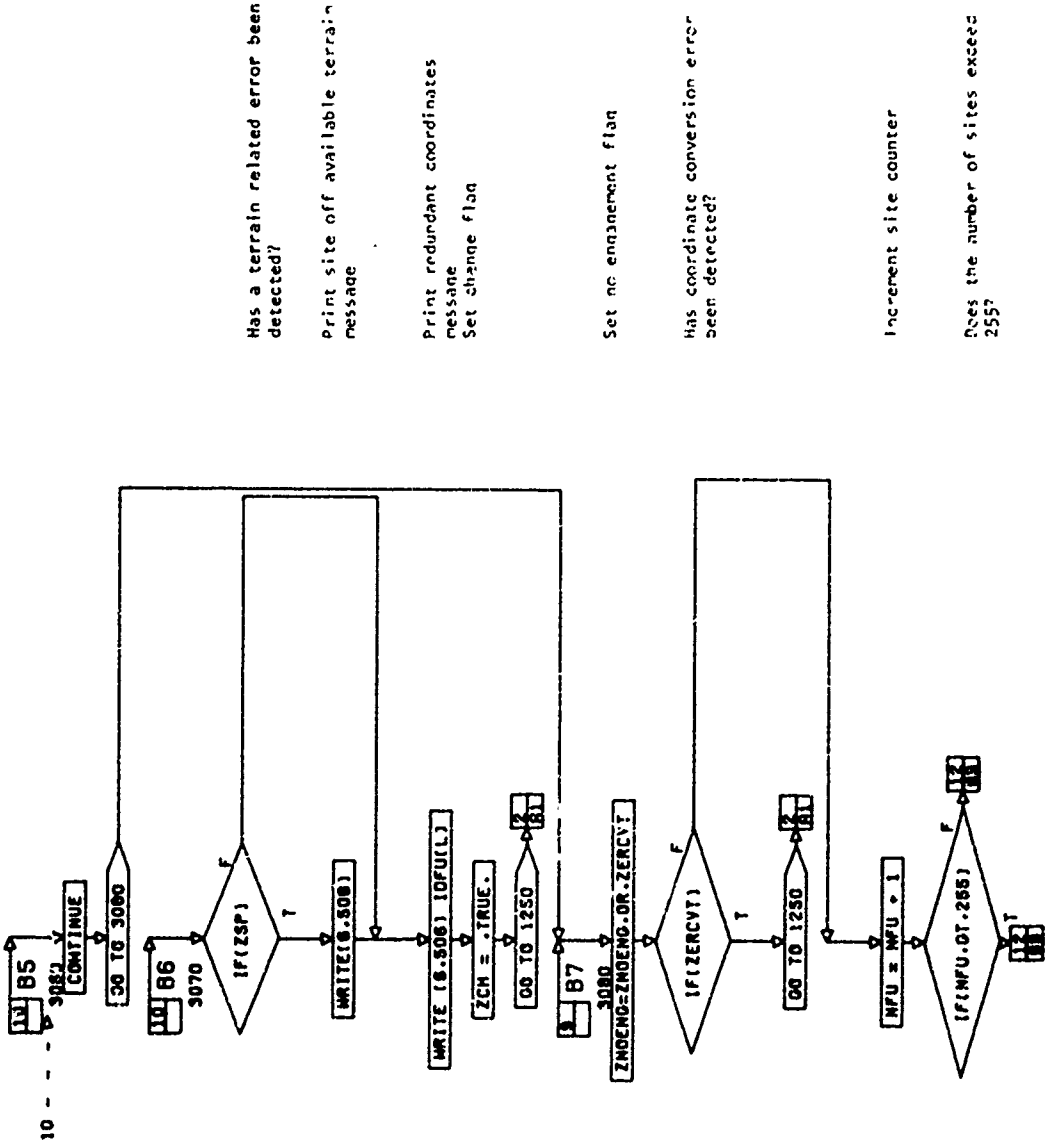
P.4.MAIN-7

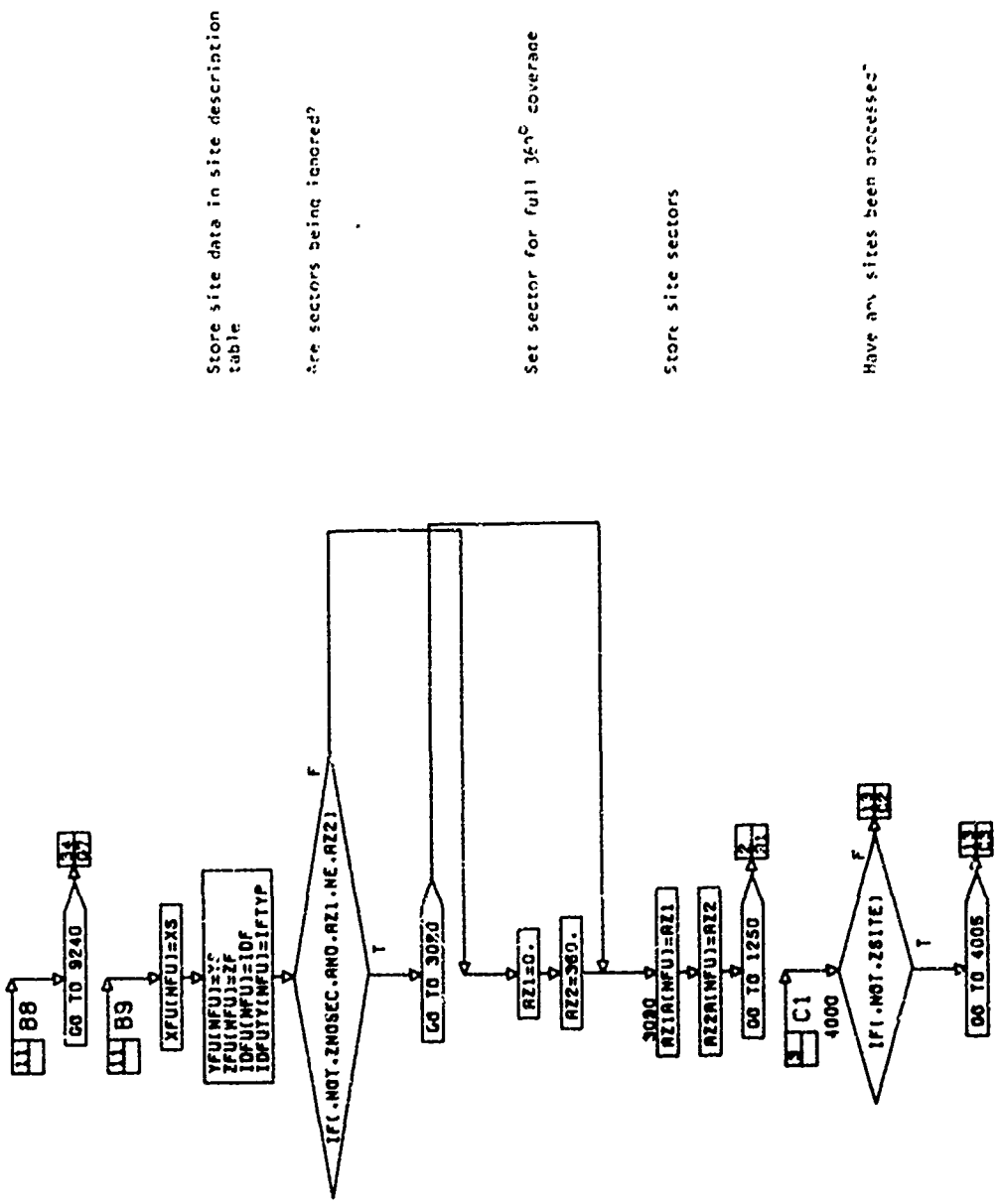

Is site table heading required?
Print site table heading
Print site description
Does this site lie on available terrain?
Print site off terrain message
Set off terrain flag
Is this site above ground level?
Print site below ground altitude message
Set below terrain flag
E
IF(INFU.EQ.0.OR.MODINFU.NLPP1.EQ.0)
T
F
WRITE(6,5111)
WRITE(6,5093,IFTYP,IOF,LTRS,NBRBCO,ZF,Z,AZ1,AZ2,XS,YS
IF(ZON)
T
F
GO TO 3018
WRITE(6,5091)
ISPa=.TRUE.
J018
IF(Z.GE.01)
T
F
GO TO 3019
WRITE(6,5101)
ISPa=.TRUE.
5

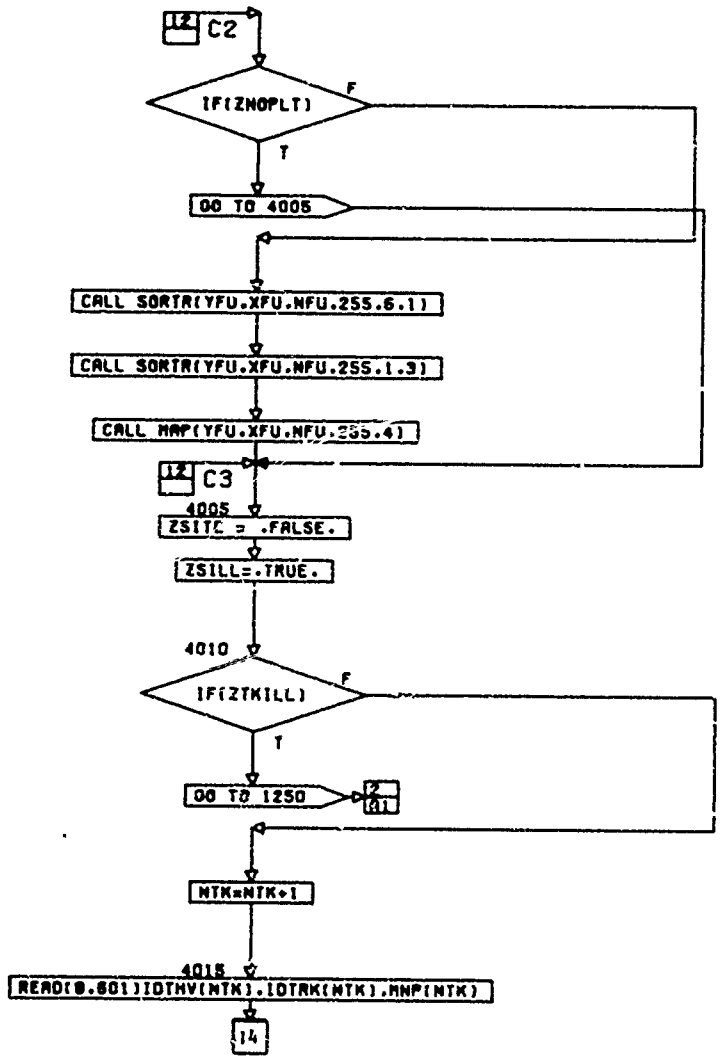
87<


Is the number of sites processed equal to zero?
DO over processed sites
Is this site name redundant?
Is this site not at the same location?
Has a terrain related error been detected?
9
A9
3019
IF(NFU.EQ.0)
F
T
GO TO 3000
DO 3020 L=1,NFU
IF(IOF.EQ.IOFUIL))
GO TO 3030
3020 CONTINUE
GO TO 3050
3030
IF(XFUIL).NE.XS.OR.YFUIL).NE.YS)
GO TO 3040
IF(ZSP)


Skip to next line
Print dunlicate card message
Set change flag
Store character 'ZIP'
Increment dunlicate site IF counter
Has a terrain related error been detected?
Skip a line
Reset terrain error flag
Print site rename message and set change flag
DO over sites
Is the site location redundant?
B1
WRITE(6,500)
B2
WRITE (6,504) L
ZCH = .TRUE.
DO TO 1250
B3
3040
ZIDF(I)=ZUID(NO)
NO=NO+1
IF(ZSP)
F
T
WRITE(6,508)
ZSP=.TRUE.
WRITE (10,506) L,IDF
ZCH = .TRUE.
B4
3050
DO 3080 L=1,NFU
- - - - D 11
IF(XFU(L).EQ.XS.AND.YFU(L).EQ.YS)
F
T
GO TO 3070


Has a terrain related error been detected?
Print site off available terrain message
Print redundant coordinates message
Set change flag
Set no engagement flag
Has coordinate conversion error been detected?
Increment site counter
Does the number of sites exceed 255?
B5
3062
CONTINUE
GO TO 3060
B6
3070
IF(ZSP)
WRITE(6,506)
WRITE(6,506) IOFU(L)
ZCH = .TRUE.
GO TO 1250
B7
3080
ZNOENG=ZNOENG.OR.ZERCVT
IF(ZERCVT)
GO TO 1250
NFU = NFU + 1
IF(NFU.GT.255)
T
F


Store site data in site description table
Are sectors being ignored?
Set sector for full 360° coverage
Store site sectors
Have any sites been processed
B8
GO TO 9240
B9
XFU(NFU)=XS
YFU(NFU)=YS
ZFU(NFU)=ZF
IDFU(NFU)=IDF
IDFUTYP(NFU)=IFTYP
IF(.NOT.ZNOSEC.AND.AZ1.NE.AZ2)
F
T
GO TO 3020
AZ1=0.
AZ2=360.
3020
AZ1A(NFU)=AZ1
AZ2A(NFU)=AZ2
GO TO 1250
C1
4000
IF(.NOT.ZSITE)
F
T
GO TO 4005



Is site plotting not desired?

Sort sites according to descending northing

Find maximum and minimum

Plot site map

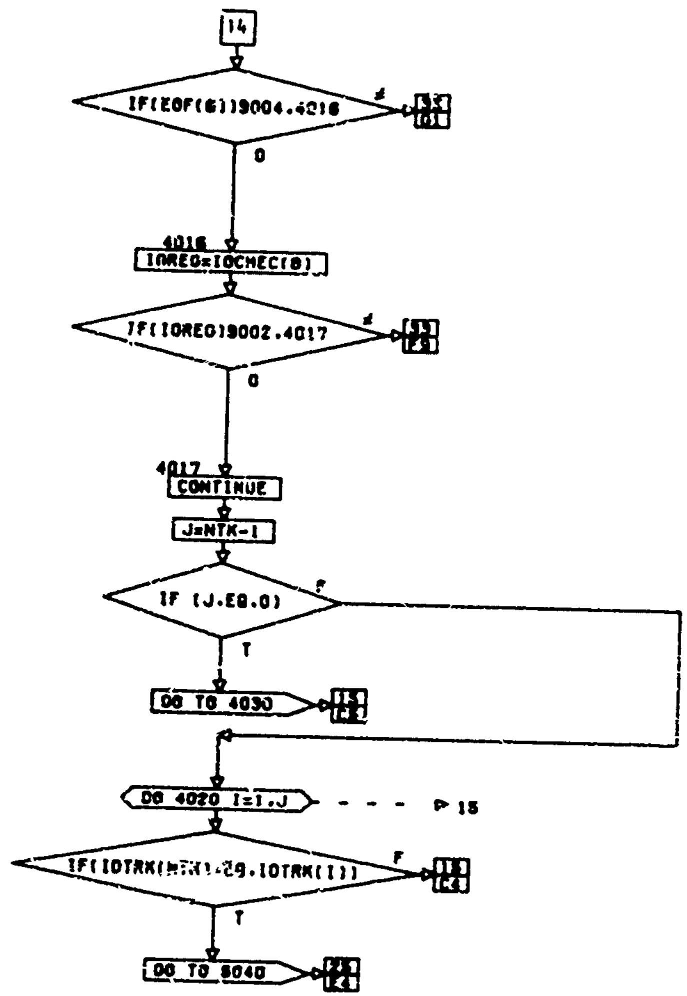
Reset site flag

Set site plot completion flag

Is kill flag TRUE?

Increment track counter

Read second TRACK card from file 8



Is EØF index zero?

Store I/Ø parity error index

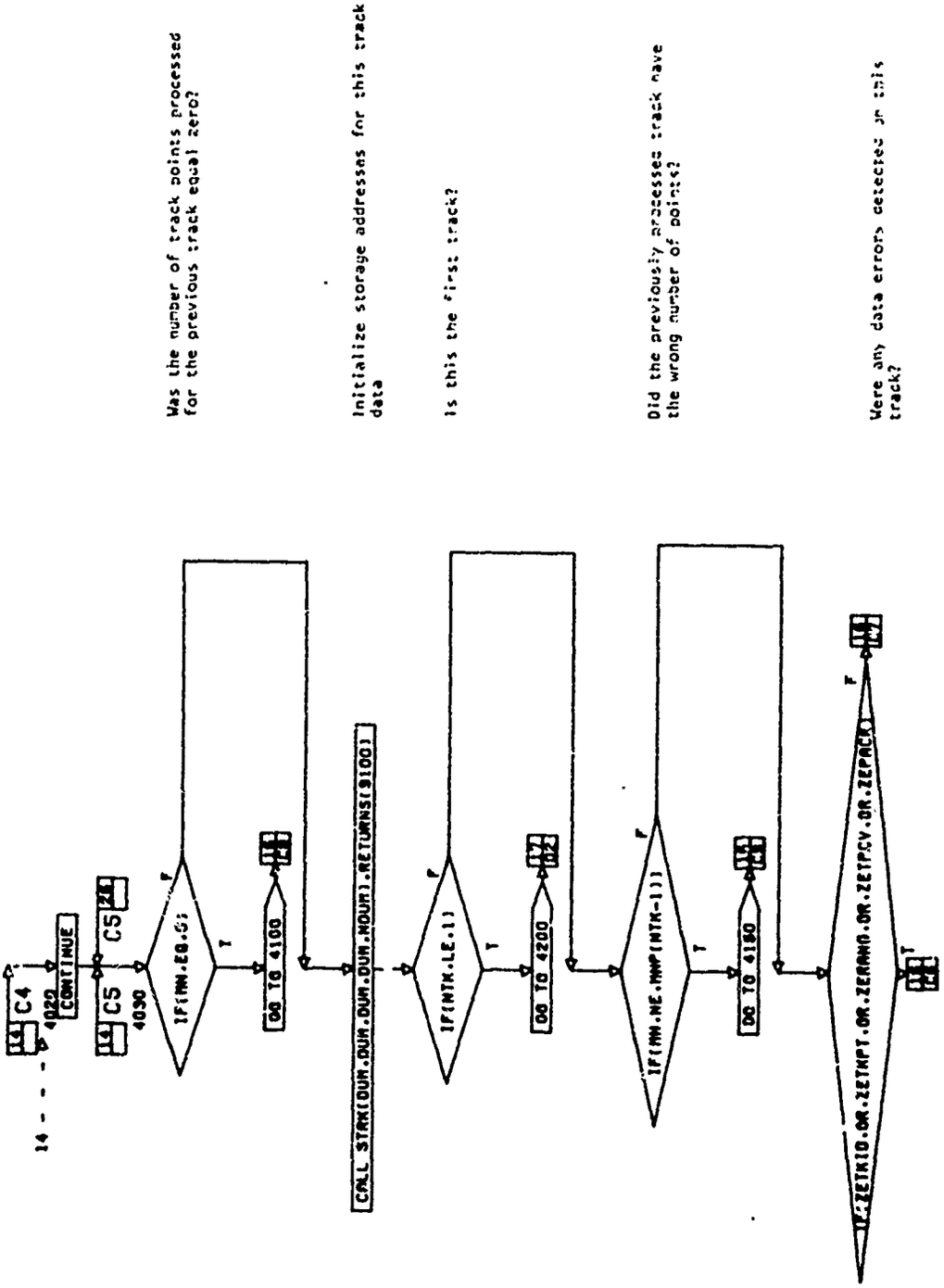
Is I/Ø parity error index zero?

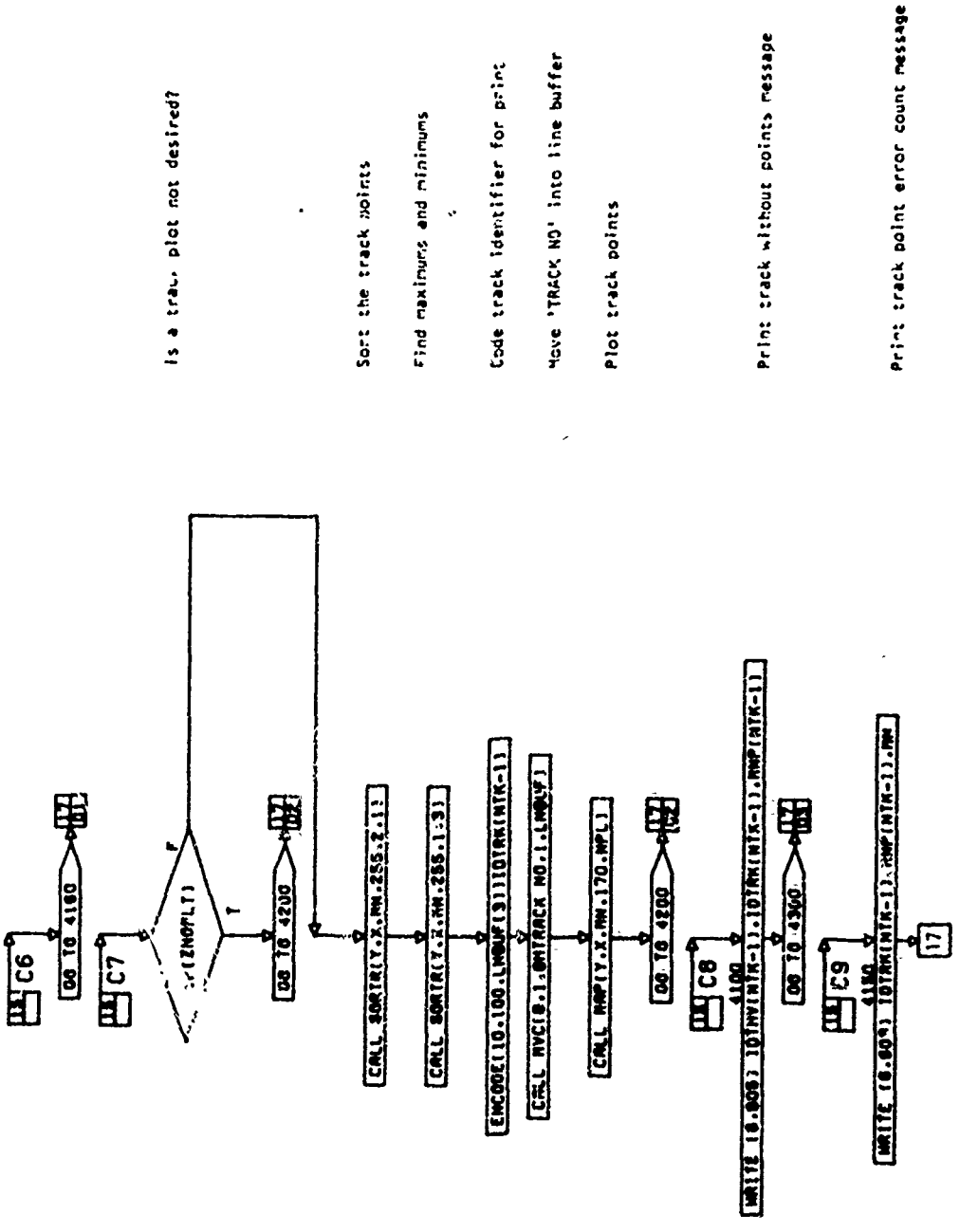
Store previous track number

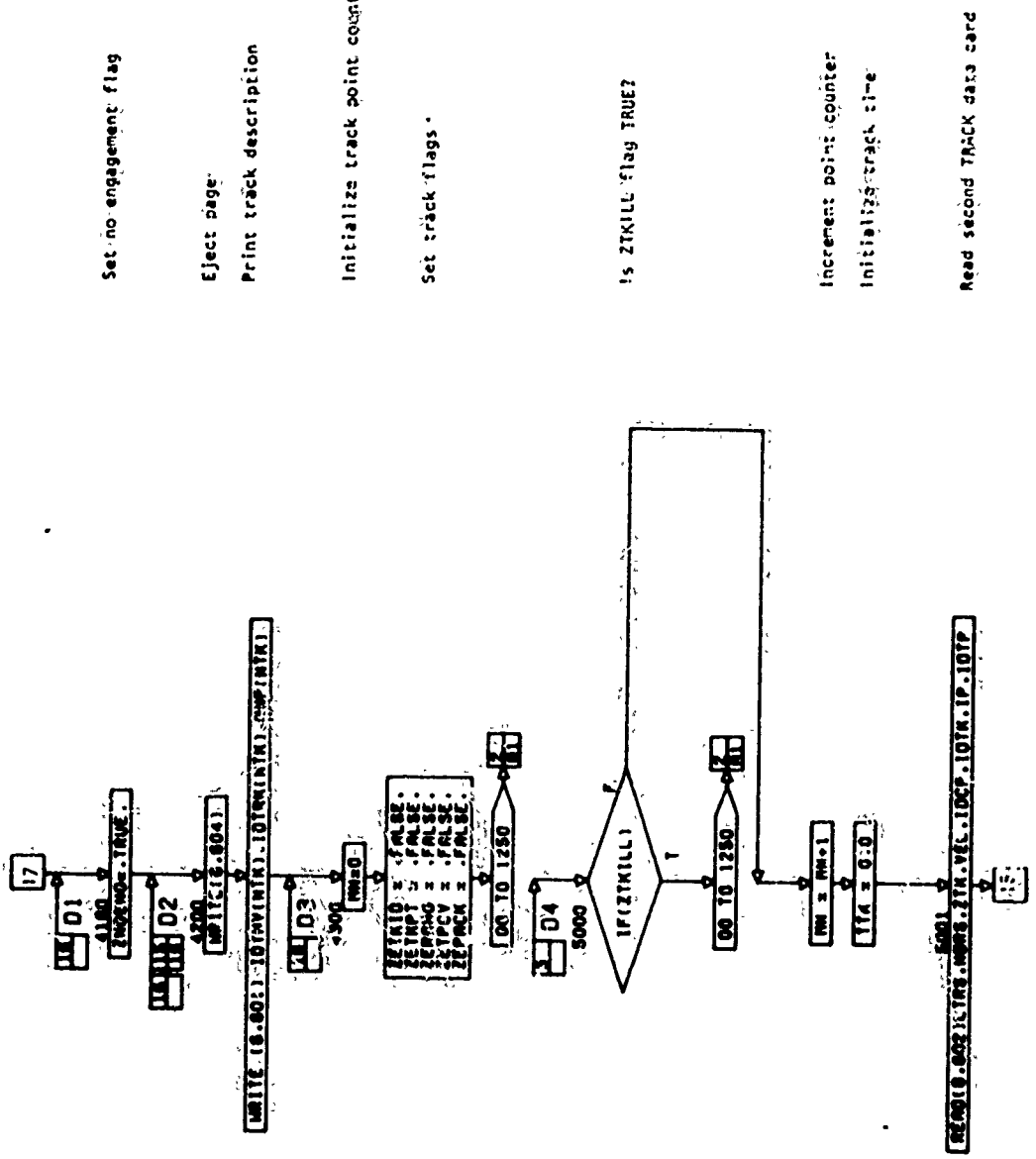
Is this the first track?

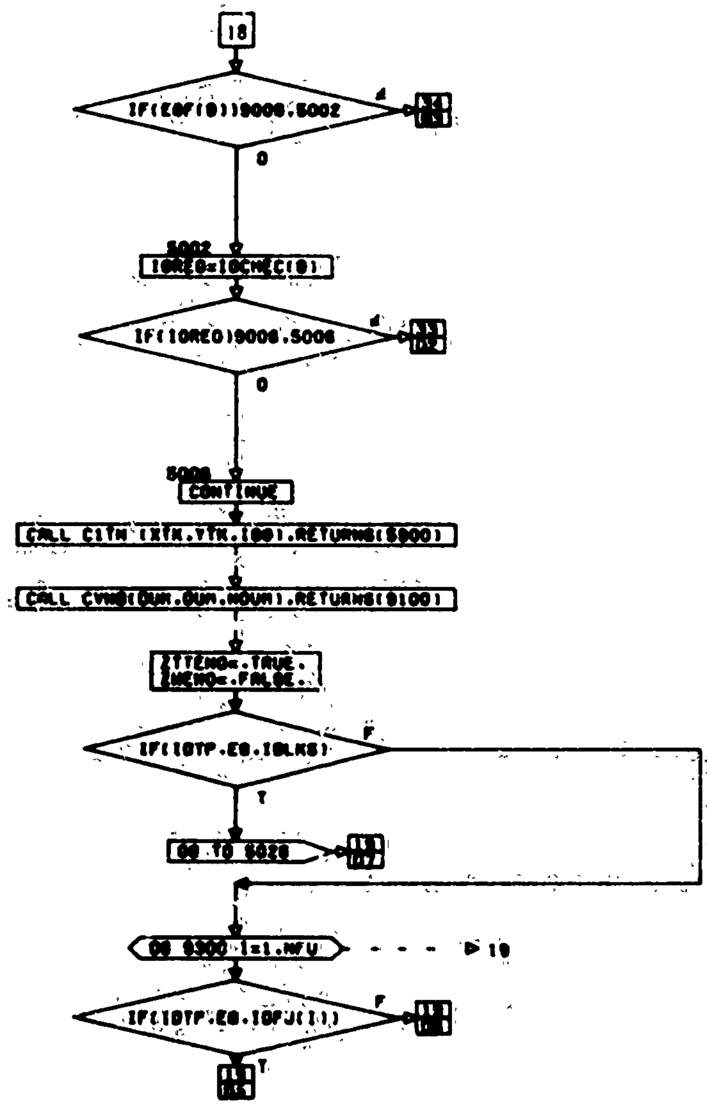
DO over all previous tracks

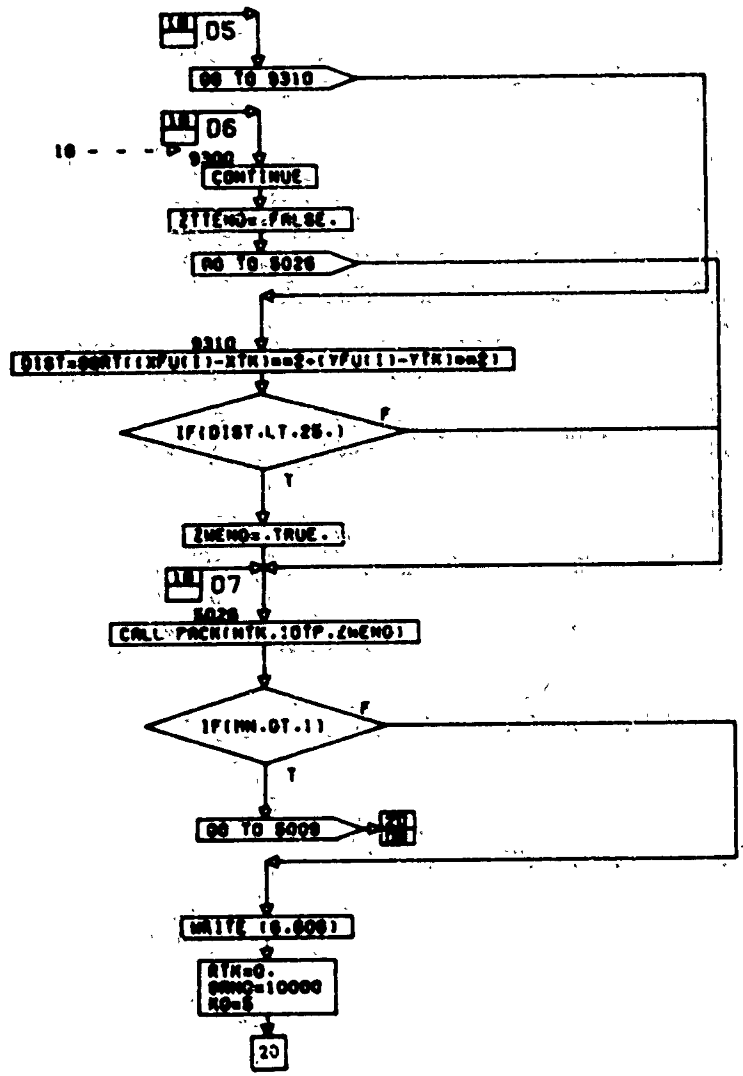
Is this track identifier redundant?


Was the number of track points processed for the previous track equal zero?
Initialize storage addresses for this track data
Is this the first track?
Did the previously processed track have the wrong number of points?
Were any data errors detected on this track?
14 - - - C4 4022
CONTINUE
14 CS 4030
CS
IF(NW.EQ.0)
T
F
GO TO 4100
CALL STAK(DUM,DUM,DUM,DUM,NOUT),RETURNS(9100)
IF(NTK.LE.1)
GO TO 4200
IF(NW.NE.NWP(NTK-1))
GO TO 4150
IF(ZETKIO.OR.ZETAPT.OR.ZERANG.OR.ZETPCV.OR.ZEPACK)


Is a trav. plot not desired?
Sort the track points
Find maximums and minimums
Code track identifier for print
Move 'TRACK NO' into line buffer
Plot track points
Print track without points message
Print track point error count message
C6
GO TO 4180
C7
F
T
GO TO 4200
CALL SORTR(Y,X,MN,255,2,1)
CALL SORTR(Y,X,MN,255,1,3)
ENCODE(10,100,LMBUF(13))TRK(NTK-1)
CALL MVC(8,1,BMTRACK NO.1,LMBUF)
CALL MAP(Y,X,MN,170,MPL)
GO TO 4200
C8
4100
WRITE (6,806) TOTMV(NTK-1),TOTRK(NTK-1),RMP(NTK-1)
GO TO 4360
C9
4180
WRITE (6,809) TOTRK(NTK-1),RMP(NTK-1),MN
17


Set no engagement flag
Eject page
Print track description
Initialize track point counter
Set track flags
Is ZTKILL flag TRUE?
Increment point counter
Initialize track time
Read second TRACK data card
I7
D1
D2
D3
D4
5000
IF(ZTKILL)
T
GO TO 1250
GO TO 1250
NM=0
NM = NM+1
TTK = 0.0


18
IF(EOF(8))9006,5002
0
5002
IOREO=IOCHEC(8)
IF(IOREO)9006,5006
0
5006
CONTINUE
CALL CITM (XTM,YTM,IGD),RETURNS(5900)
CALL CVNO(DUM,DUM,NOUM),RETURNS(9100)
ITTEMO=.TRUE.
IMEMO=.FALSE.
IF(IDTP.EQ.IBLKS)
F
T
GO TO 5020
DO 9300 I=1,NFU
19
IF(IDTP.EQ.IDFU(I))
F
T
Is EØF index equal zero?
Store I/Ø parity error index
Is I/Ø parity error index zero?
Convert point location to UTM
Code grid coordinates for print
Set track target flags
Is this point identifier blank
DO over sites
Is this track targeted for this site?



Set no target test flag

Calculate point/site distance

Is the miss distance less than 25 meters?

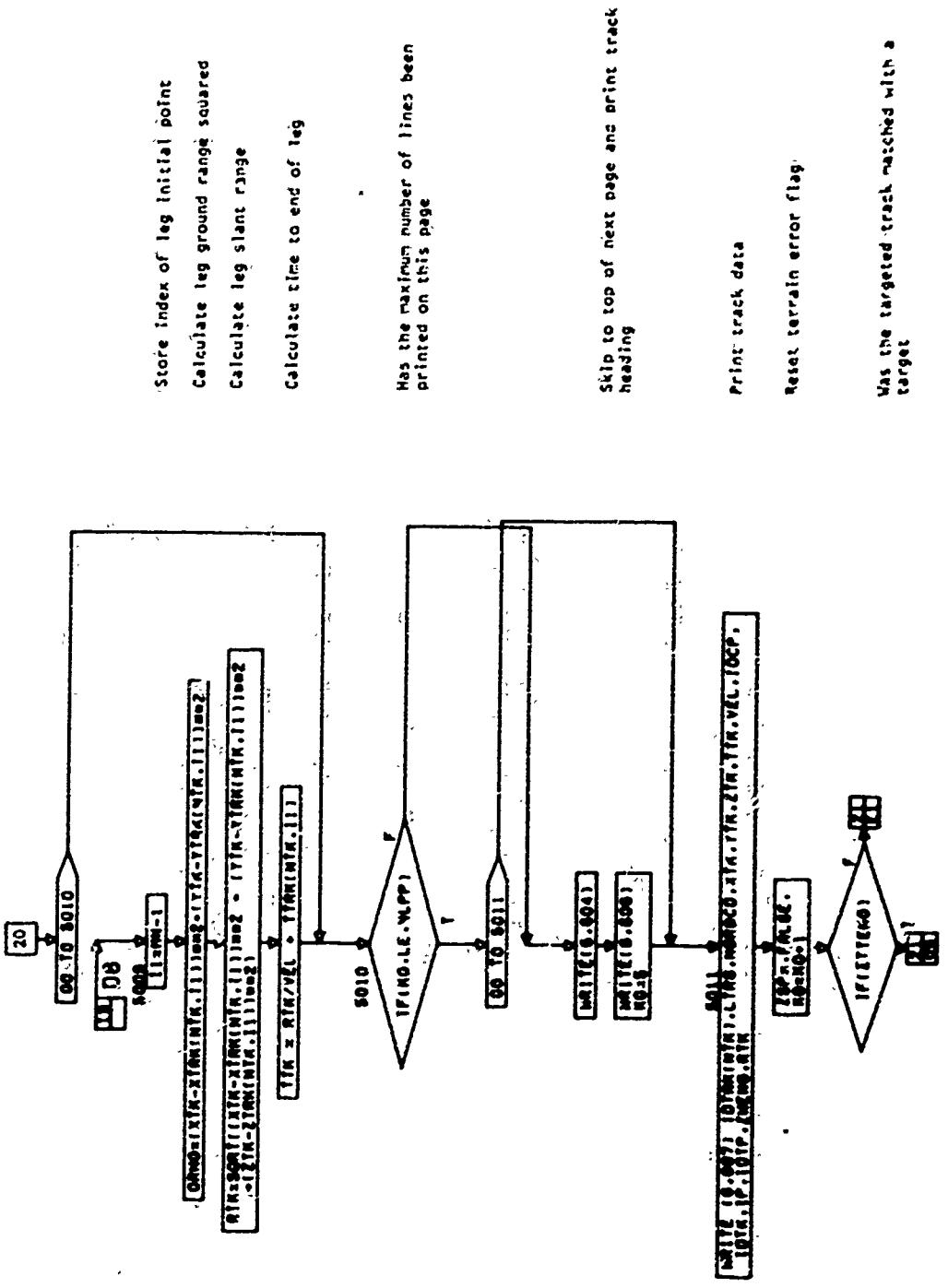
Set target flag

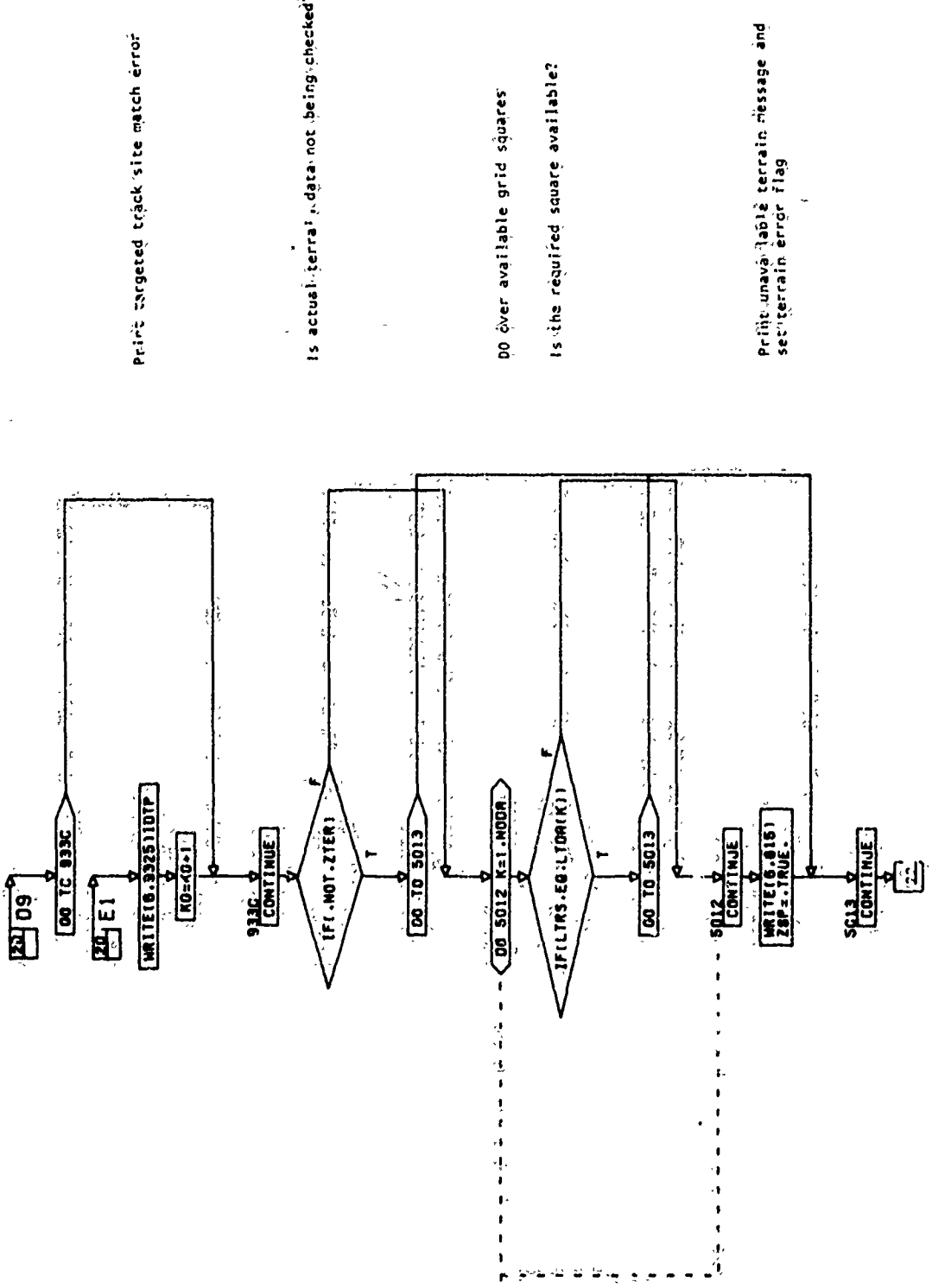
Store track data

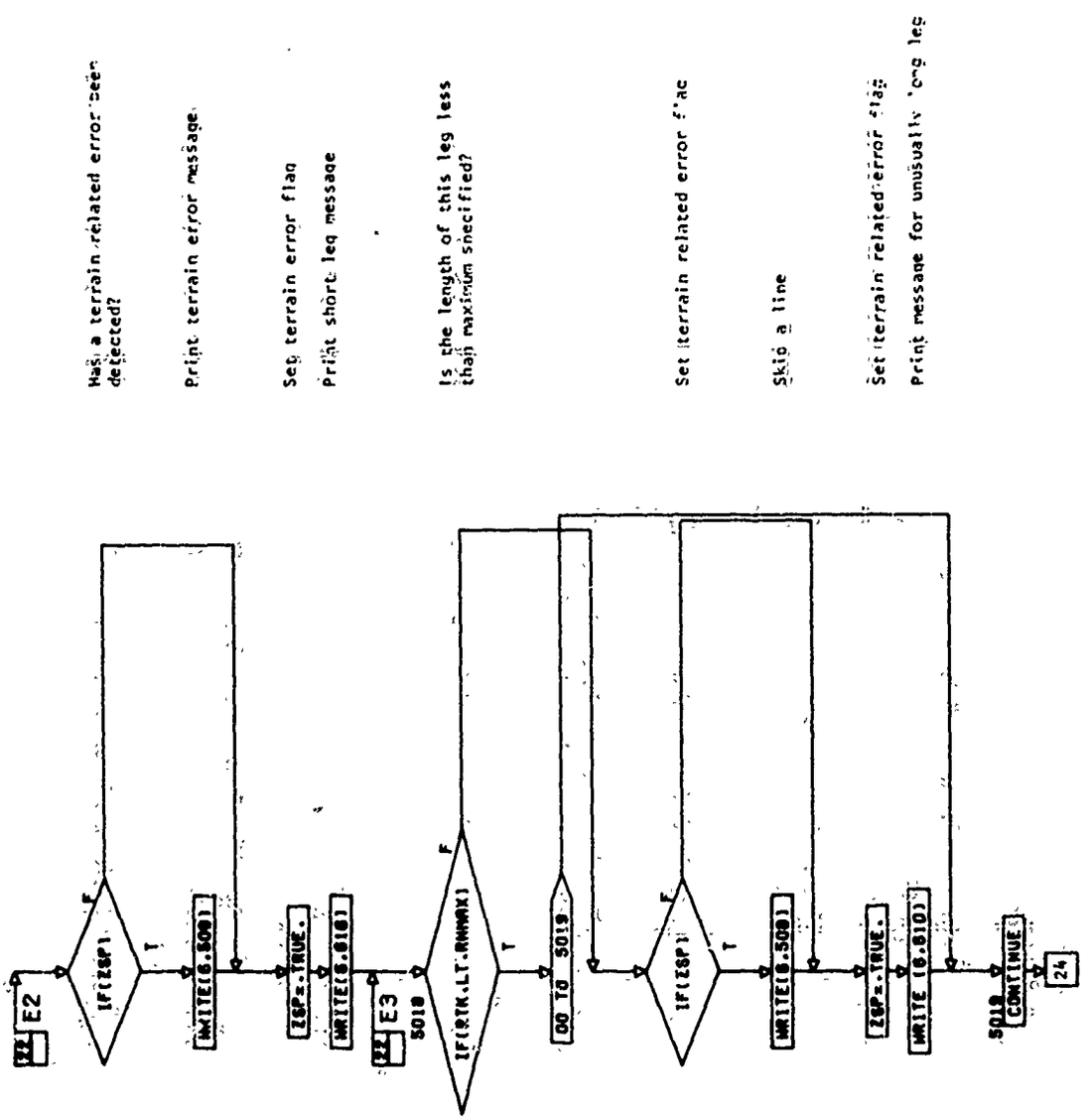
Have other points been processed for this track?

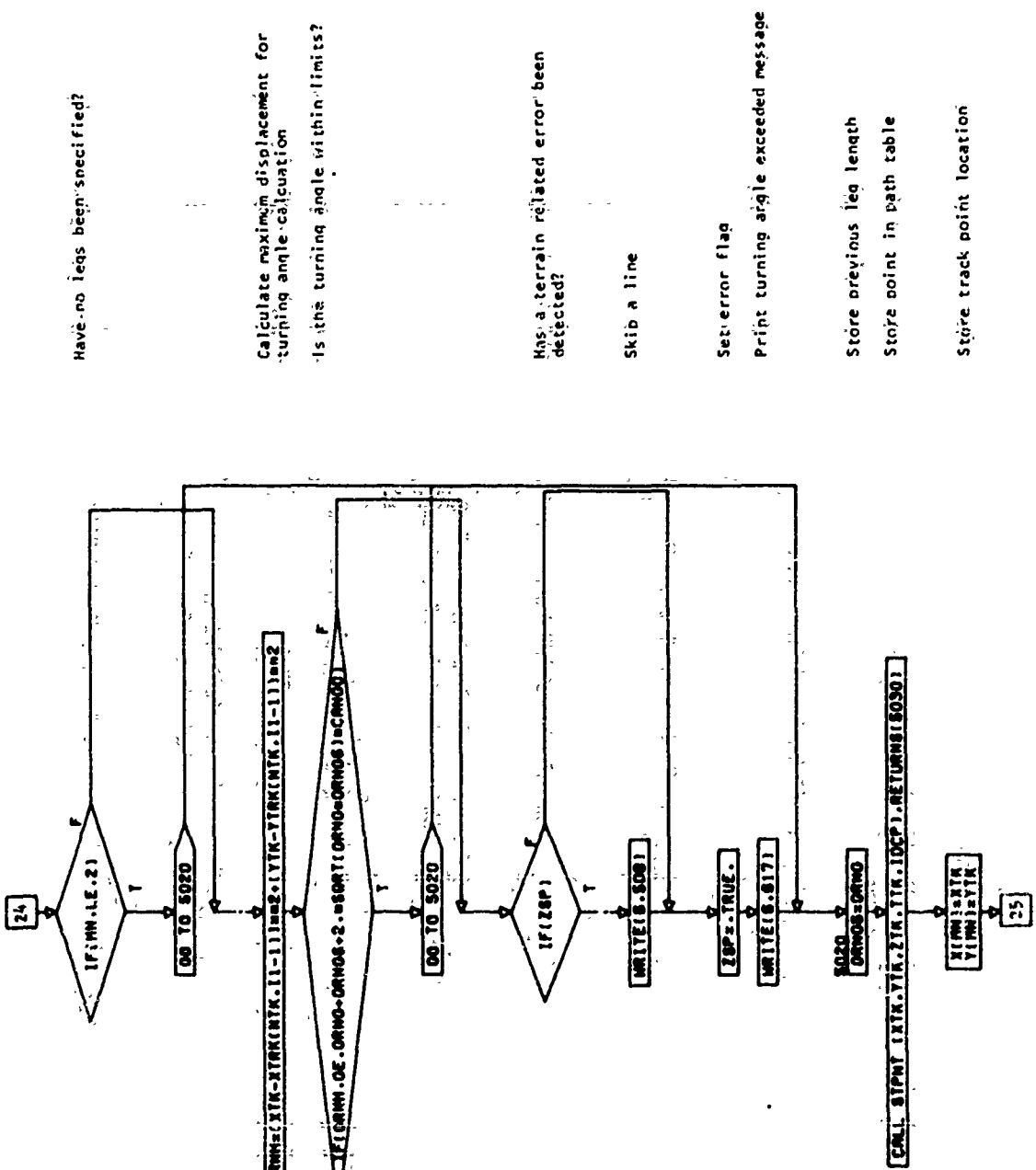
Print track point headings

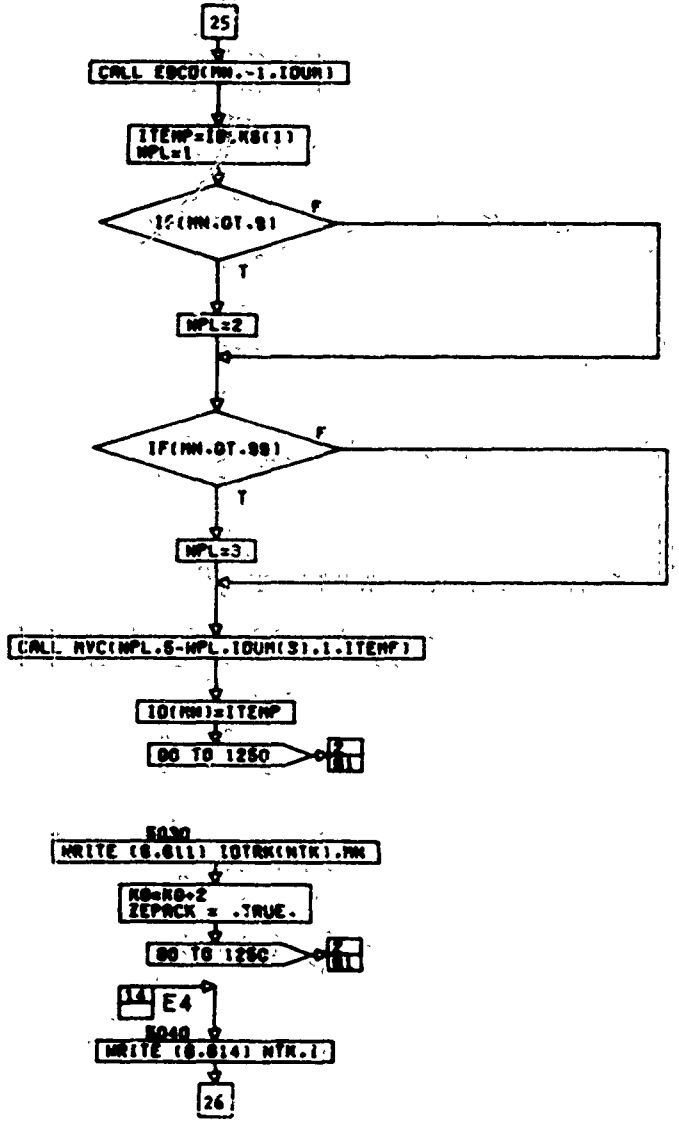
Initialize track leg data


Store index of leg initial point
Calculate leg ground range squared
Calculate leg slant range
Calculate time to end of leg
Has the maximum number of lines been printed on this page
Skip to top of next page and print track heading
Print track data
Reset terrain error flag
Was the targeted track matched with a target
20
GO TO 8010
$010
IF(NO.LE.MPP)
GO TO 8011
IF(ITEST(60)


Print targeted track site match error
Is actual terrain data not being checked?
DO over available grid squares
Is the required square available?
Print unavailable terrain message and set terrain error flag
20 D9
DO TC 933C
20 E1
WRITE(6,3325)IDTP
KO=KO+1
933C CONTINUE
IF(.NOT.ZTER)
F
T
GO TO 5013
DO 5012 K=1,NOOR
IF(LTRS.EQ.LTOR(K))
F
T
GO TO 5013
5012 CONTINUE
WRITE(6,815) ZBP=.TRUE.
5013 CONTINUE


Has a terrain related error been detected?
Print terrain error message
Set terrain error flag
Print short leg message
Is the length of this leg less than maximum specified?
Set terrain related error flag
Skip a line
Set terrain related error flag
Print message for unusually long leg
E2
IF(ISP)
WRITE(6,500)
ISP=.TRUE.
WRITE(6,810)
E3
5010
IF(STK.LT.RMMAX)
GO TO 5019
IF(ISP)
WRITE(6,500)
ISP=.TRUE.
WRITE (6,810)
5019 CONTINUE
24
T
F


Have no legs been specified?
Calculate maximum displacement for turning angle calculation
Is the turning angle within limits?
Has a terrain related error been detected?
Skip a line
Set error flag
Print turning angle exceeded message
Store previous leg length
Store point in path table
Store track point location
24
IF(NN.LE.2)
T
F
GO TO 5020
ORNN=(XTK-XTRK(NTK,II-1))**2+(YTK-YTRK(NTK,II-1))**2
IF(ORNN.GE.ORNO*ORNO+2.*SQRT(ORNO*ORNO)*CRNOO)
T
F
GO TO 5020
IF(ZSP)
T
F
WRITE(6,5001)
ZSP=.TRUE.
WRITE(6,5171)
5020
ORNO=ORNN
CALL STPNT(XTK,YTK,ZTK,TTK,IOCP),RETURN(16090)
XTRN=XTK
YTRN=YTK
25



Code point number for print

Initialize point identifier

Does the point sequence number require more than one digit?

Set number of sequence digits

Does the point sequence number require more than two digits?

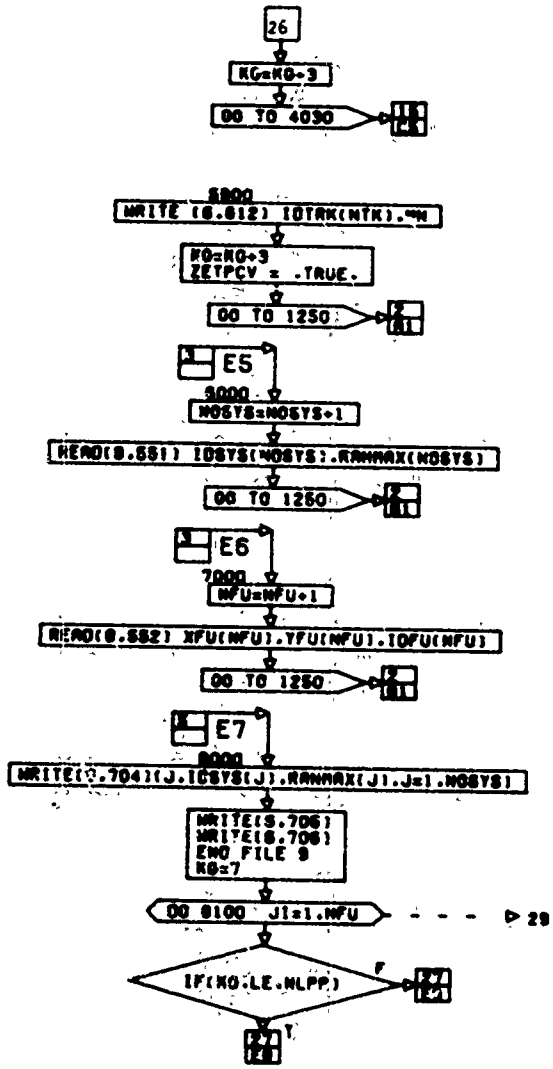
Set number of sequence digits

Store sequence number

Print pack failure message

Set pack error flag

Print redundant track error message



Increment line counter

Print conversions message

Set conversion error flag

Increment system counter

Read second SYSTEM type card from file 3

Increment site counter

Read second SITE type card from file 8

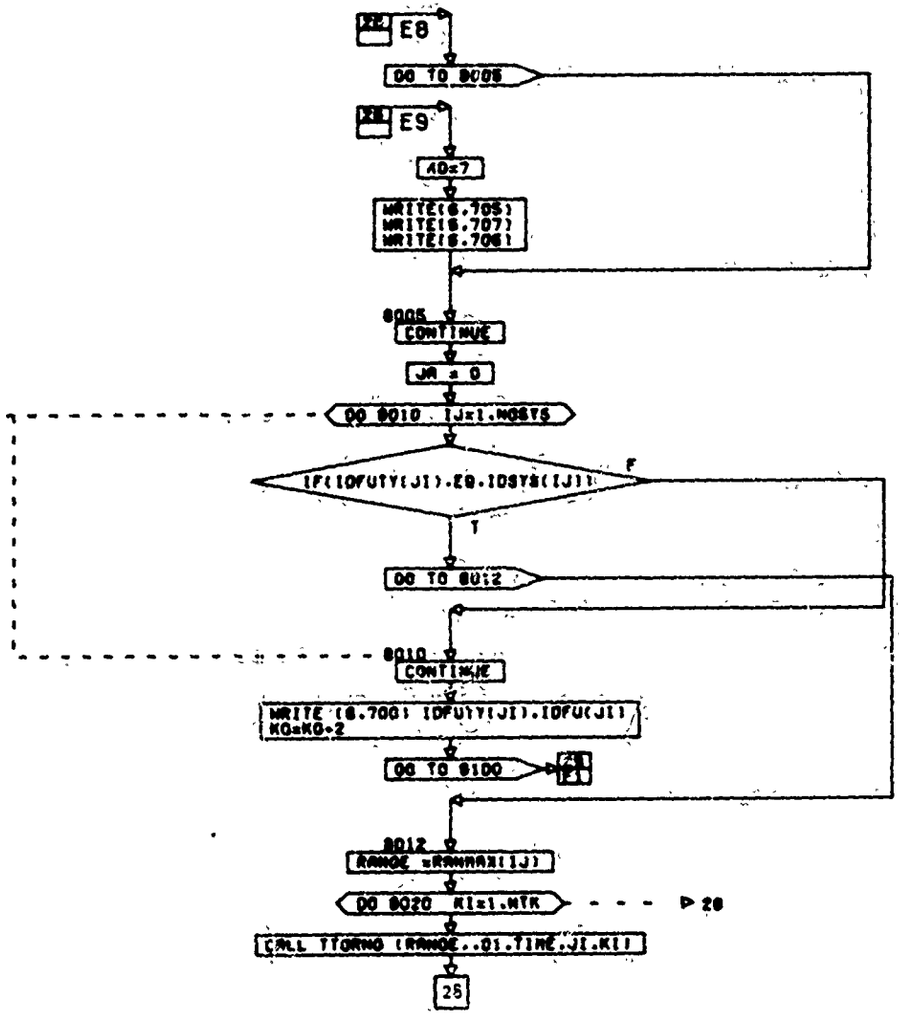
Print system data table

Print engageability table header

End file 9

DO over sites

Is space left for print on this page?



Print engagability table heading

DO over systems

Does this site have a dispatchable system type?

Print site system error message

Store system maximum range

DO over tracks

Calculate time of track in fire volume

Is this track not engagable by this site

Increment engagability counter

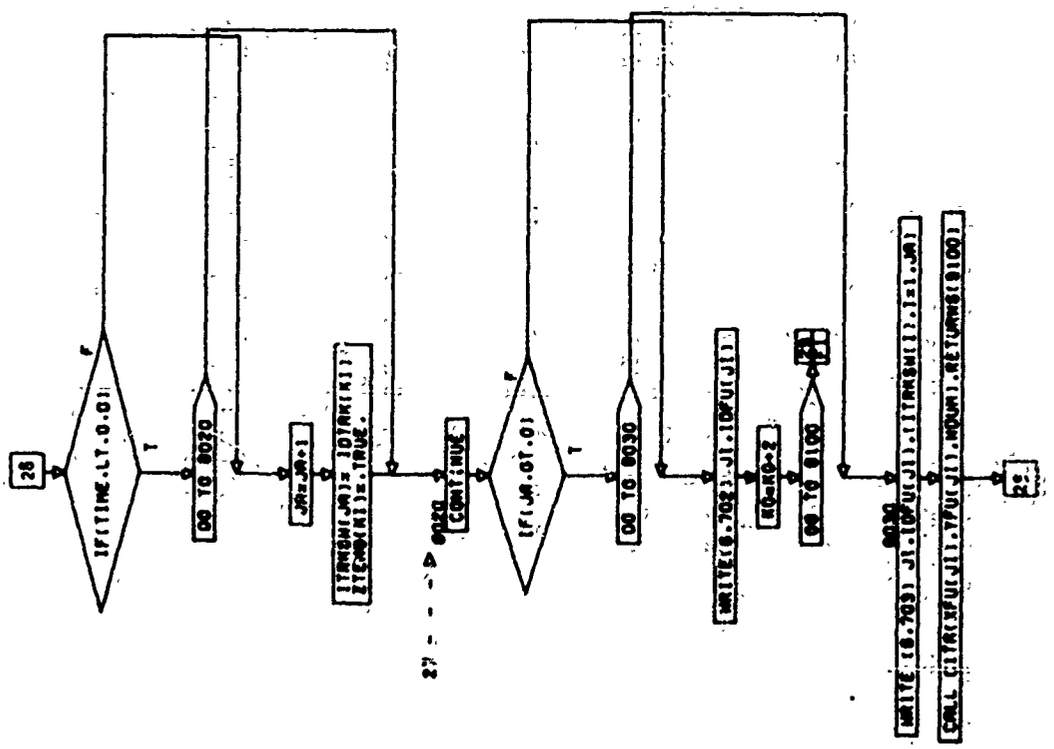
Store track identifier and set engageability flag

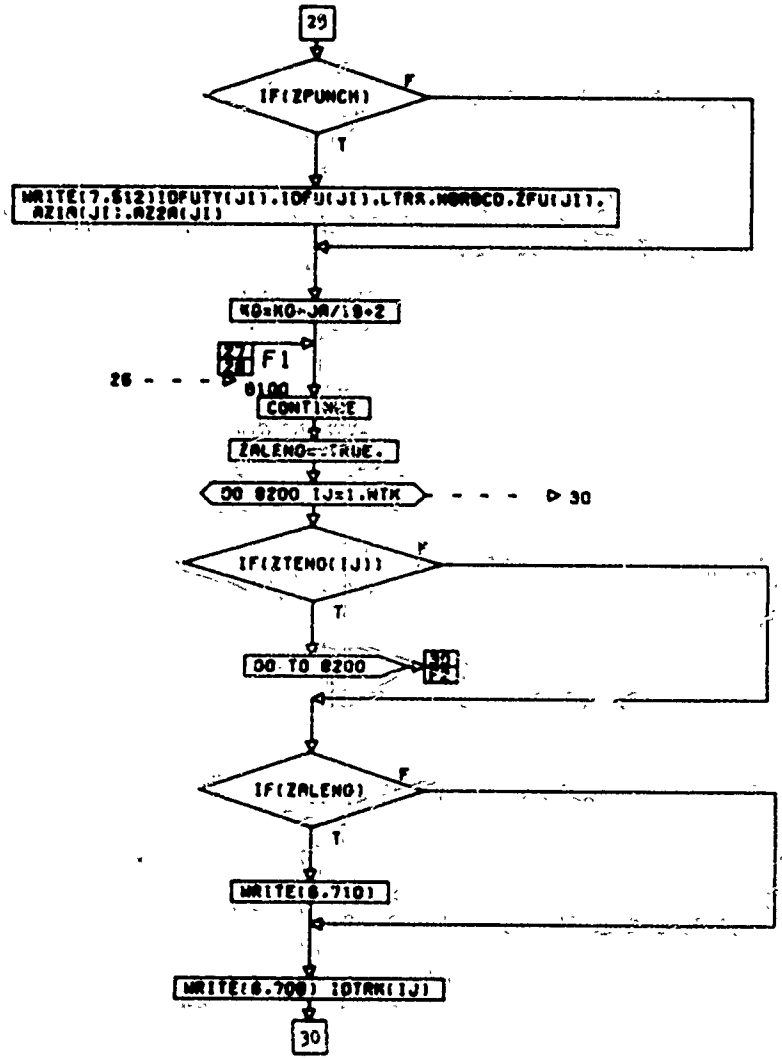
Have engageable sites been determined?

Print engageability

Print engageable tracks for this site

Convert site location to UTM





Are site cards to be punched?

Punch site cards on file 7

Increment line counter

Set engagement flag

DO over tracks

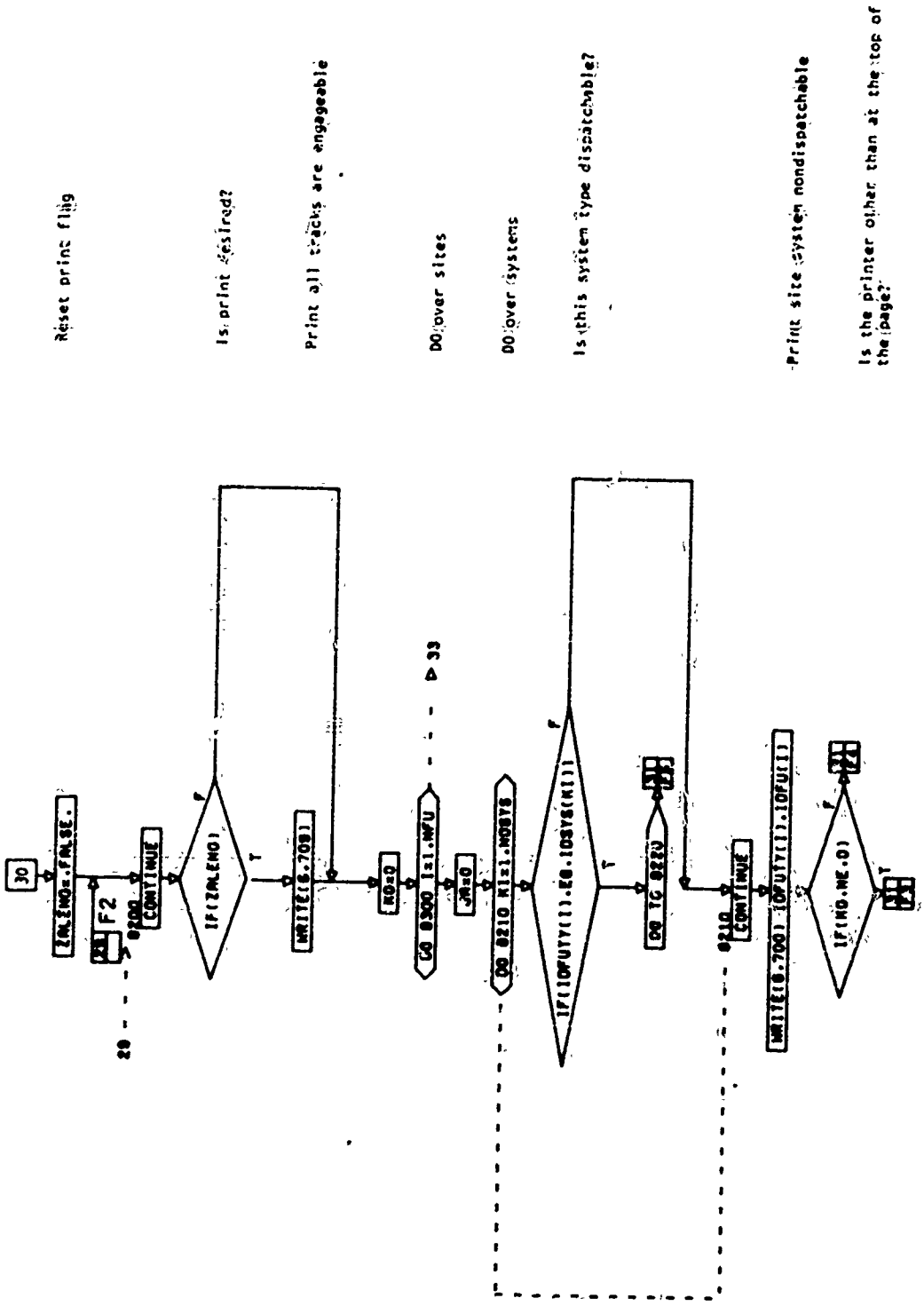
Is this track engageable?

Is this the first track?

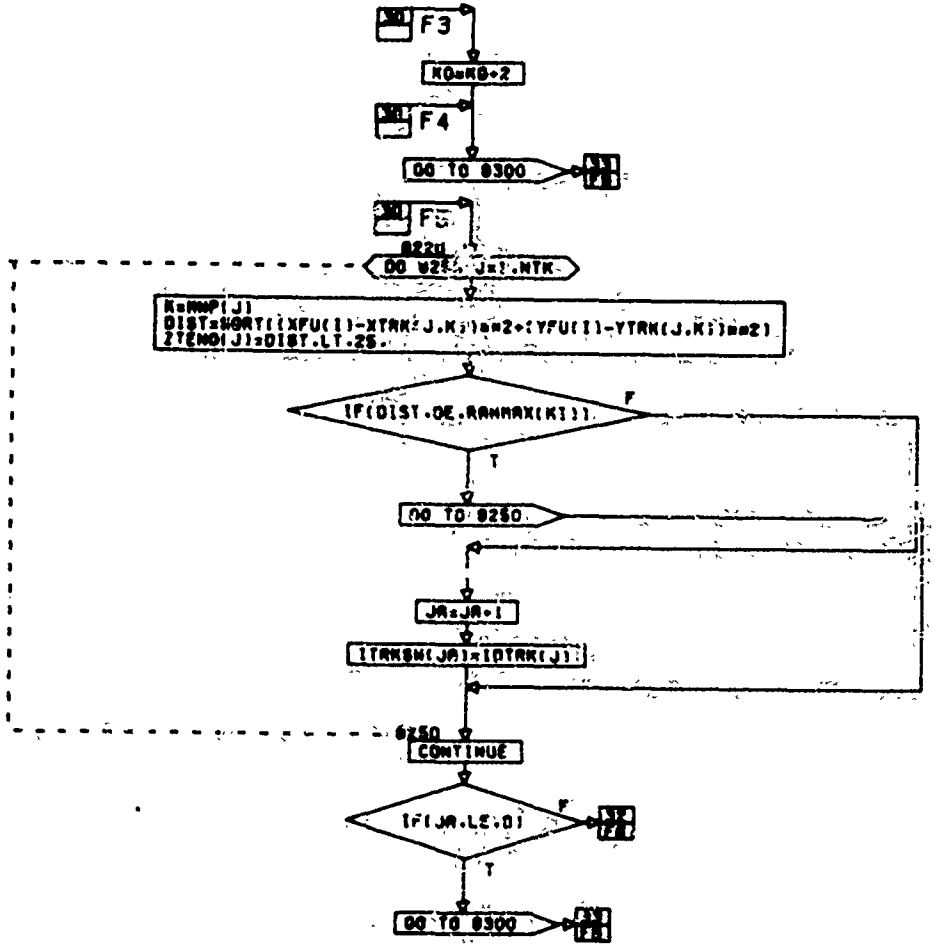
Print unengageable track message

Print track identifier

P,4.MAIN-29


Reset print flag
Is print desired?
Print all tracks are engageable
DO over sites
DO over systems
Is this system type dispatchable?
Print site system nondispatchable
Is the printer other than at the top of the page?
30
IM.ENO=.FALSE.
F2
9200 CONTINUE
IF(IM.ENO)
WRITE(6,7001)
KO=0
GO 9300 I=1,NFU
JM=0
DO 9210 K=1,NOSYS
IF(IOFUTY(I).EQ.IOSYS(K))
DO TO 9211
9210 CONTINUE
WRITE(6,7001) (OFUTY(I),IOFU(I))
IF(NO.NE.0)
20
33

P.4.MAIN-30



Increment line counter

DO over tracks

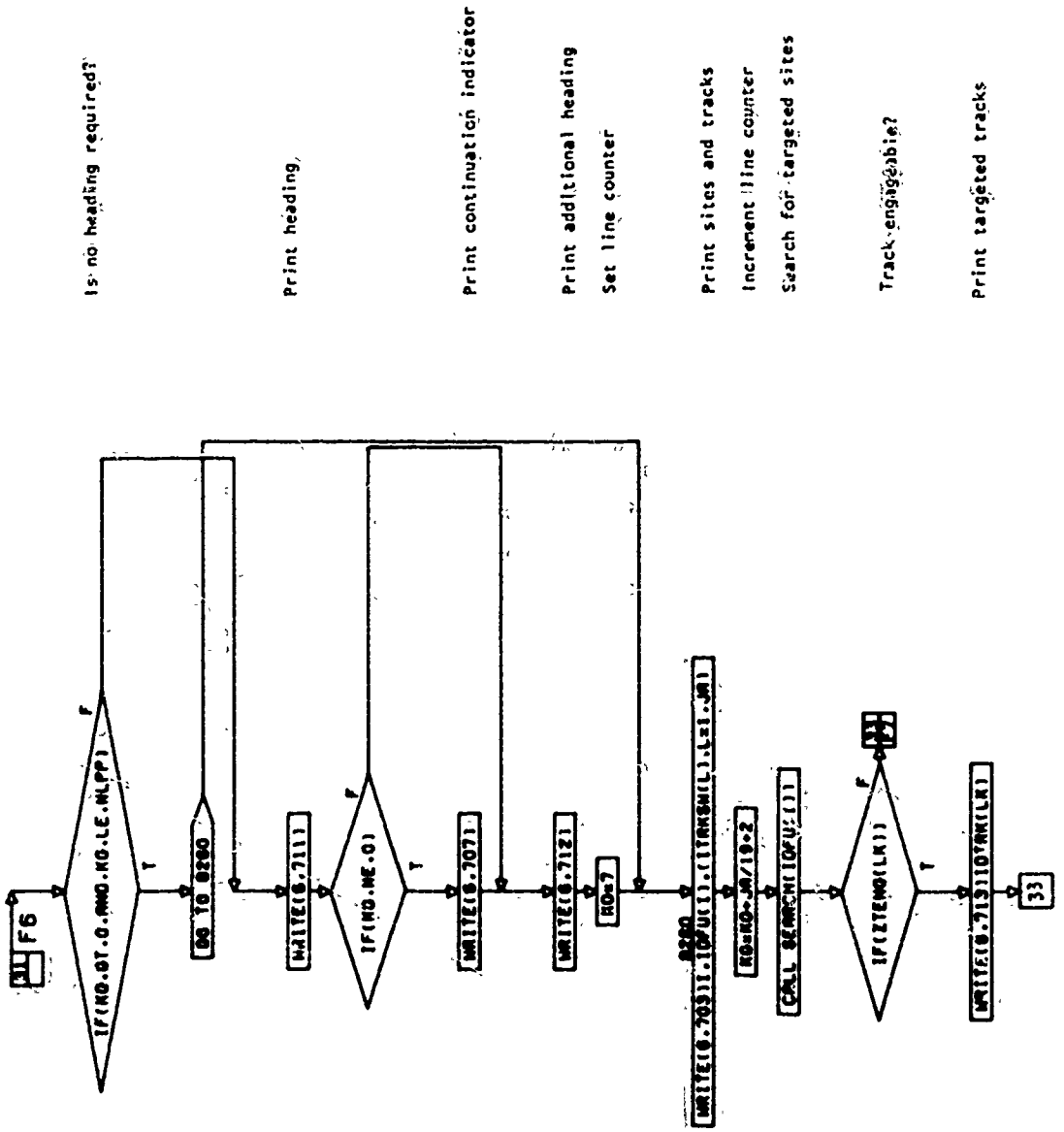
Calculate displacement between last track point and site
Set displacement flag

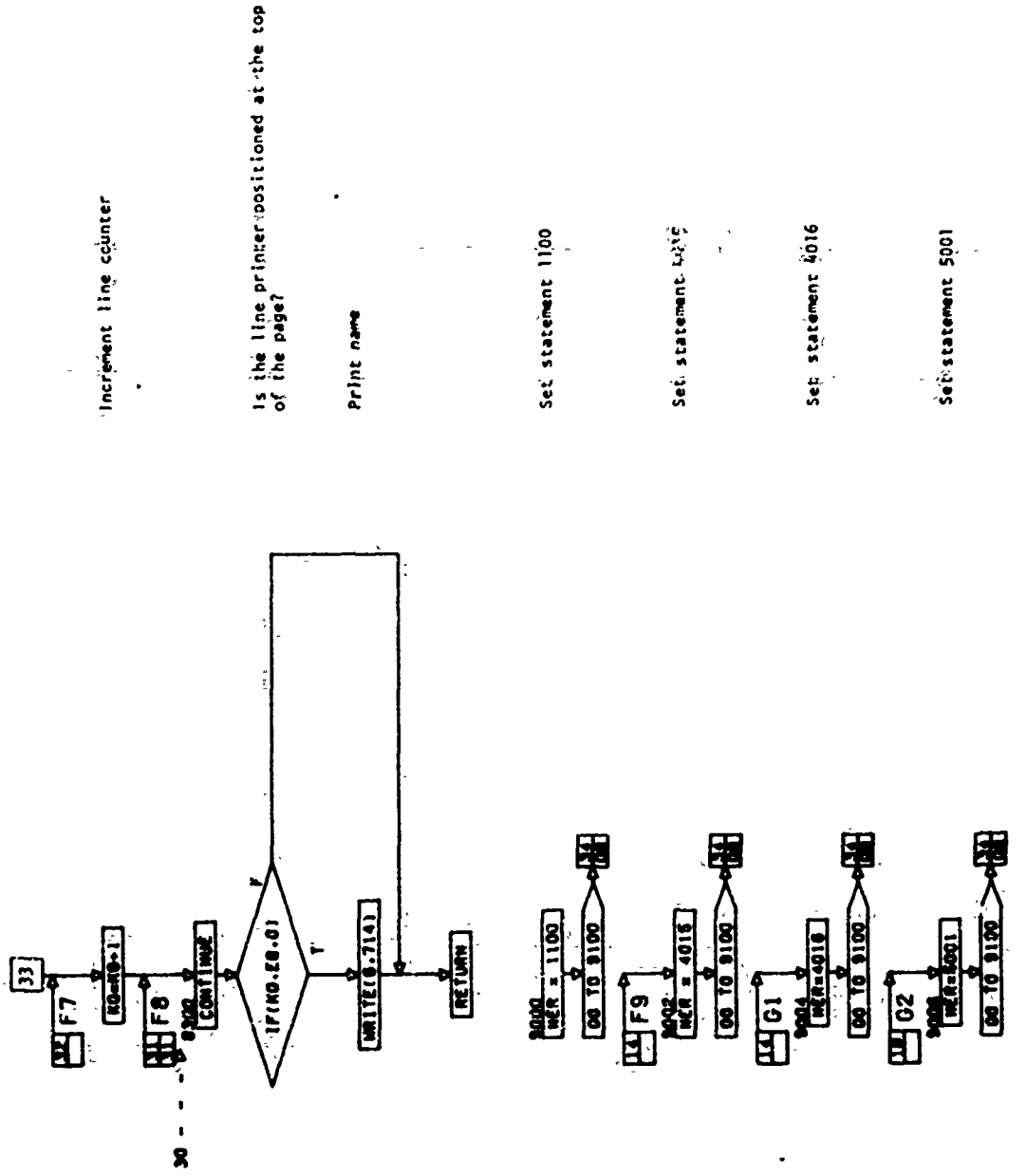
Does the track end point lie outside the fire volume?

Increment fire volume counter

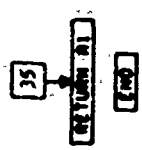
Store track identifier

Do any tracks end in a fire volume?


Is no heading required?
Print heading
Print continuation indicator
Print additional heading
Set line counter
Print sites and tracks
Increment line counter
Search for targeted sites
Track engagable?
Print targeted tracks
11 F6
IF(KO.GT.0.AND.KO.LE.NLPP)
GO TO 8200
WRITE(6,211)
IF(KO.NE.0)
WRITE(6,707)
WRITE(6,7121)
KO=7
8200
KO=KO-NP/19+2
CALL SEARCH(IOFU,I,I)
IF(ITEND(LK))
33


Increment line counter
Is the line printer positioned at the top of the page?
Print name
Set statement 1100
Set statement 4016
Set statement 5001
33
F7
NO=NO+1
F8
CONTINUE
IF(NO.EQ.0)
WRITE(6,7141)
RETURN
F9
G1
G2
MCR = 1100
MCR = 4016
MCR=5001
GO TO 9100


Set statement 5002
Set statement 1250
Set statement 3000
Set statement 3010
Print site coordinate error message
Set conversion error flag
Set terrain altitude error flag
Set statement 3080
Print error message


35
RETURN #1
END

1. NAME: MAP

2. TYPE OF PROGRAM: SUBROUTINE, OUTPUT

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Print a map containing the points of interest.

5. ASSUMPTIONS AND LIMITATIONS: The coordinates of the points of interest have been appropriately loaded into the arrays in the proper form.

6. ERROR RETURNS: A call to ERR and a RETURN A1 will be executed if either of the following errors are detected:

   A. 2000 - ABSCØR is unable to generate the required abscissa coordinate labels.

   B. 2900 - LABEL is unable to generate the ordinate labels.

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL MAP (Y, X, NY, MAXNY, NCH), RETURNS (A1)

   Y - Array, relative northing of the points of interest.
   X - Array, relative easting of the points of interest.

   NY - Number of points to be printed on the map.

   MAXNY - Maximum number of points to be printed.

   NCH - Maximum number of characters in site identifier.

   A1 - Exit taken if one of the conditions listed under 6(above) occurs.

8. PROGRAMS CALLING THIS PROGRAM: MAIN

9. COMMON VARIABLES USED:

   A. ZZCØRD - DELTA, ESTEDG, IØRBUF, LNBUF (equivalenced to ZLNBUF), WSTEDG

   B. ZZMAPP - NLPI, SCALE, XMAX, XMIN, YMAX, YMIN

   C. ZZCV1 - YØRG

10. COMMON VARIABLES TO BE SET:

A. ZZCØRD - IØRBUF, LNBUF (equivalenced to ZLNBUF)

B. ZZMAPP - NLPI, SCALE, XMAX, XMIN, YMAX, YMIN

C. ZZCVI - YØRG

11. COMMON VARIABLES CHANGED:

A. ZZCØRD - DELTA, ESTEDG, LNBUF (equivalenced to ZLNBUF), WSTEDG

B. ZZMAPP - XMAX, YMIN

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

A. SUBROUTINES

(1) CALL ABSCØR (ISTART, ISPC, N), RETURNS (A1)

ISTART - Integer, the starting position for the abscissa coordinates in the print array.

ISPC - The number of blanks between coordinate labels.

N - Number of labels to be printed across the page.

A1 - Control transfers to this statement number if an error is detected in ABSCØR.

(2) CALL ERR (NAME, LAST, NSTAT) - Prints an error message indicating name of subroutine and statement number where error occurred.

NAME - Name of subroutine where error was detected.

LAST - End address of the storage area to be dumped from subroutine NAME.

NSTAT - Statement number in subroutine NAME where error occurred.

(3) CALL LABEL (M, X, XORG, ID), RETURNS (A1)

M - Character, contains the one letter three digit grid label for the line X. .

X - Relative easting (ID=E) or northing (ID=N) for which the UTM grid letter is desired.

XØRG - Displacement (meters) of the origin from the UTM reference point (AA).

ID - Character, E indicates easting of strip is desired. N indicates northing of line is desired.

AI - Control is transferred to this statement when an illegal grid letter is detected by CLN.

(4) CALL MVC (ICNT, INPTR, ZINADD, IØPTR, ZIØADD) - Move a character string from the word ZINADD into the word ZIØADD.

ICNT - Integer, number of characters to be moved from ZINADD to ZIØADD.

INPTR - Integer, index of the first character to be moved from ZINADD.

ZINADD - Character, 2 word array (20 character) from which characters will be moved.

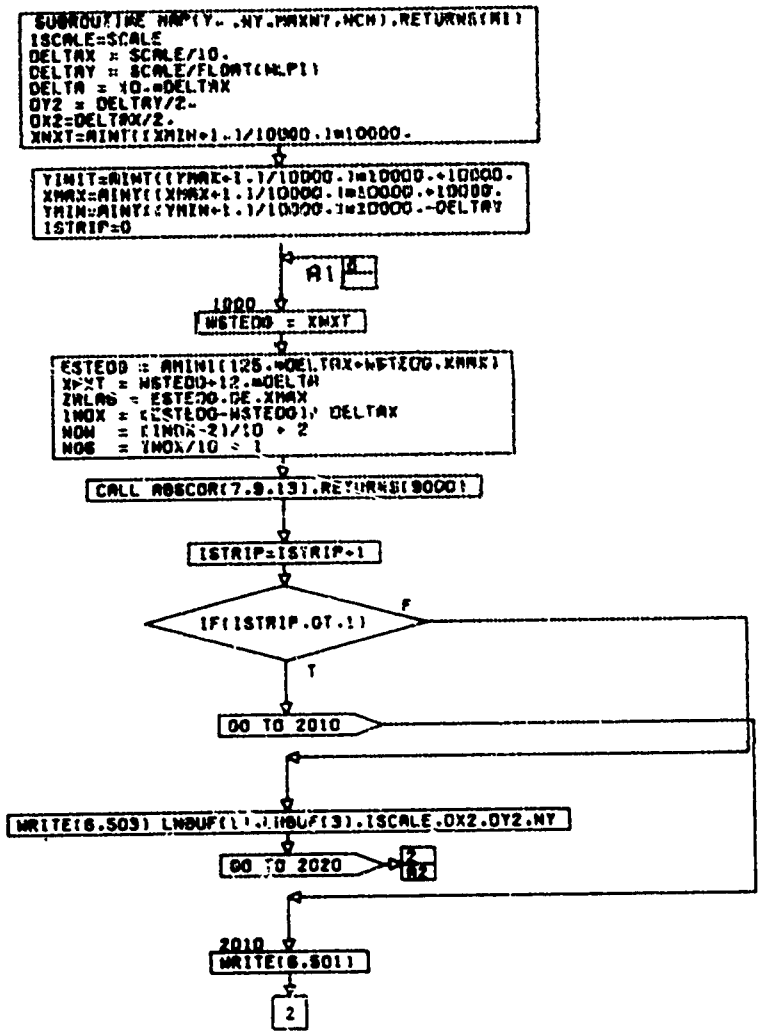
IØPTR - Integer, index of the location in ZIØADD where the first character will be moved.

ZIØADD - Character, 2 word array (20 characters) where the character will be moved.

13. NOTES OF METHODOLOGY: Upon entry, MAP sets the scale factor (ISCALE), calculates the easting scale factor (DELTAX), calculates the northing scale factor (DELTAY), calculates the ordinate step size, and calculates the map boundries. Next, the strip counter (ISTRIP), which counts the number of north-south strips required to cover the desired area map, is initialized to zero.

Next, a loop is entered which calculates and plots the north-south strips required to cover the above mentioned map. Following statement 1000, the eastern edge (ESTEDG), and western edge (WSTEDG) of the strip are calculated. The western edge of the next strip (XNXT) is calculated and the logical flag (ZRLAB) is set TRUE if no more strips are required to cover the desired map. ABSCØR is then called to store the abscissa labels for this strip in buffer IØRBUF. The strip counter (ISTRIP) is then incremented and tested. If the strip under consideration is the first strip to be processed for this map, a page eject is executed and the map header data printed. Otherwise, a skip to the top of the next page is executed. At statement 2020, the one letter three digit abscissa label is printed across the page. Next the site counter (I), the northing of the line to be printed (YAXIS) and the line counter (KG) are initialized for this strip. At statement 2100 the northern and southern boundaries of the area which this line of print represents are calculated. Next the DO loop is used to clear the line buffer. Logical flag ZLABEL then is set to TRUE if an ordinate label is required for this line. The grid for this line is next created. First the character '.' is placed in the line buffer in locations corresponding to the abscissa labels. If an ordinate label is required (ZLABEL = TRUE), the line buffer is then filled to print a complete line of grid points. At statement 2500, the site counter is then tested. If all the sites have been previously mapped, control transfers to statement 2800. Following statement 2500, logical flag ZØUTRG is set TRUE, if site I does not lie on the area represented by this line and logical flag ZCØINC is set TRUE if the next site in the site table

(I+1) lies on this area. If this point (I) does not lie on this area, control transfers to statement 2800. Otherwise, a test is performed to determine if the point I lies within the strip. If this point lies within this strip and within this line area, the point location (LØC) in the line buffer is then calculated. If this location lies outside the printer page an error message is printed and LØC is reset to 1. Following statement 2600 the line counter is incremented by seven to provide the line offset, the number of characters identifying the point is calculated and stored in the line buffer in the appropriate location for this point. The point counter is then incremented and logical flag ZCØINC is tested to determine if another point lies on this line area. If it is determined that another point is to be printed on this line, control transfers to statement 2500. A test is then performed to determine if an ordinate label is required for this line. If a label is required for this line, LABEL is then called to find this axis label and MVC is used to store this label in the line buffer. This line is then printed on the map and the northing is indexed to the next line, and control is transferred to statement 2100 if printing for this strip is not complete. Otherwise, the abscissa labels are printed on the end of the map and control returns to statement 1000 if it is required to process another strip. Otherwise, the normal return is executed to the calling program.



Set map scale
Calculate abscissa scale meters/space
Calculate ordinate scale meters/line
Calculate ordinate step size
Calculate abscissa step size
Find western boundary

Find northern boundary
Find eastern boundary
Find southern boundary
Initialize step counter

Store the western edge

Find the eastern edge
Calculate next western edge
Set last strip flag

Set grid loop indices

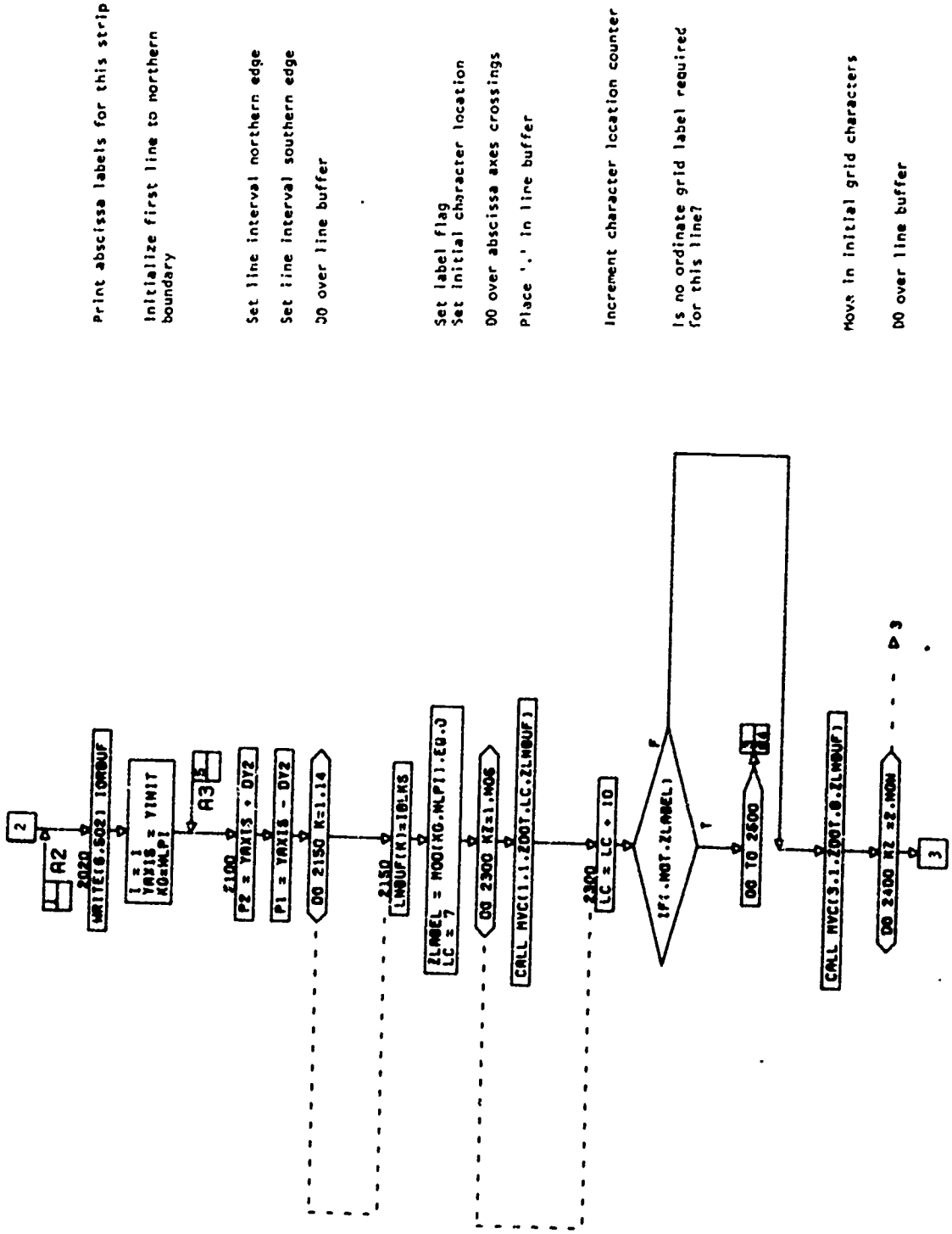
Calculate abscissa labels for this strip

Increment strip counter

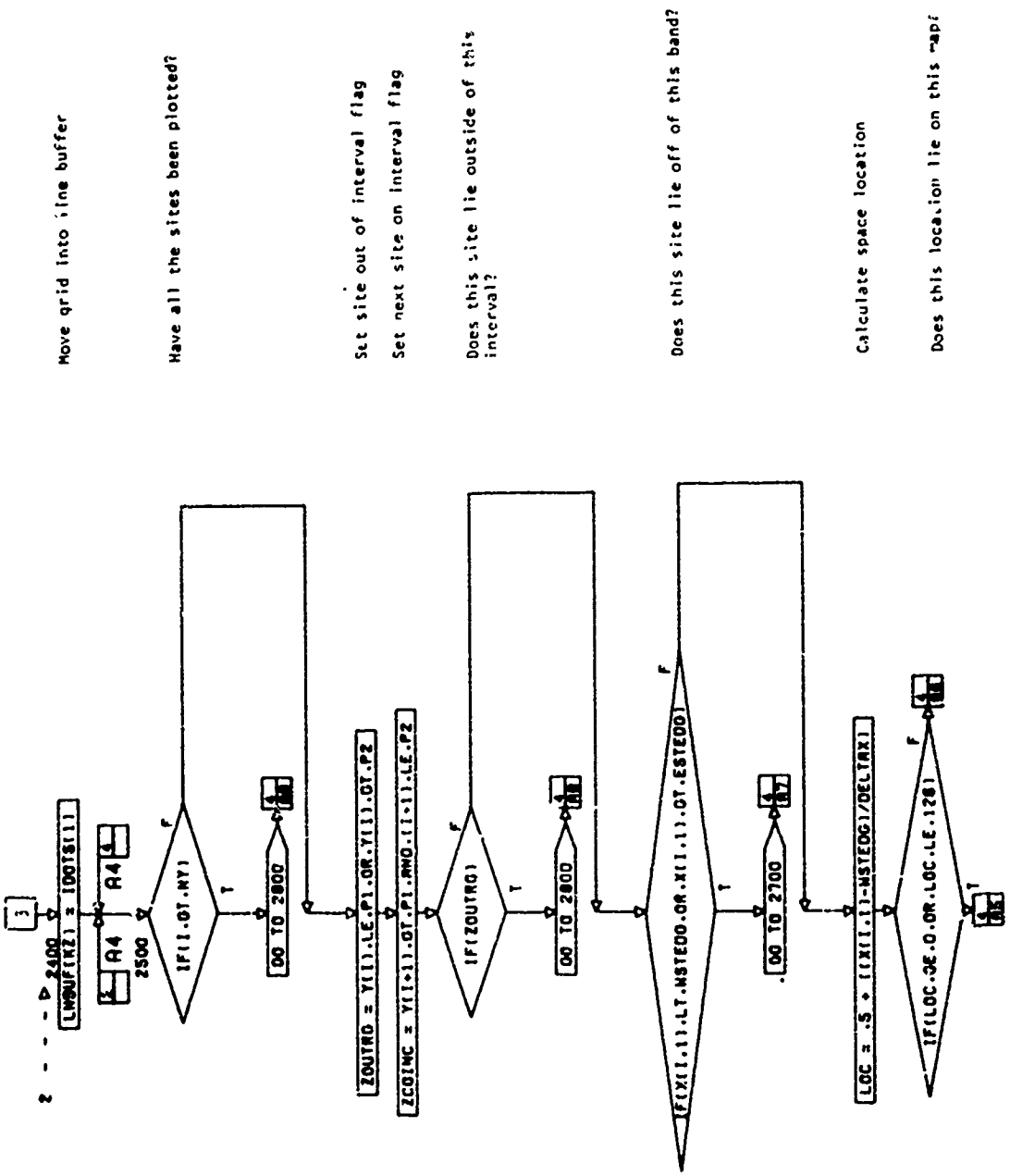
Is this other than the first strip

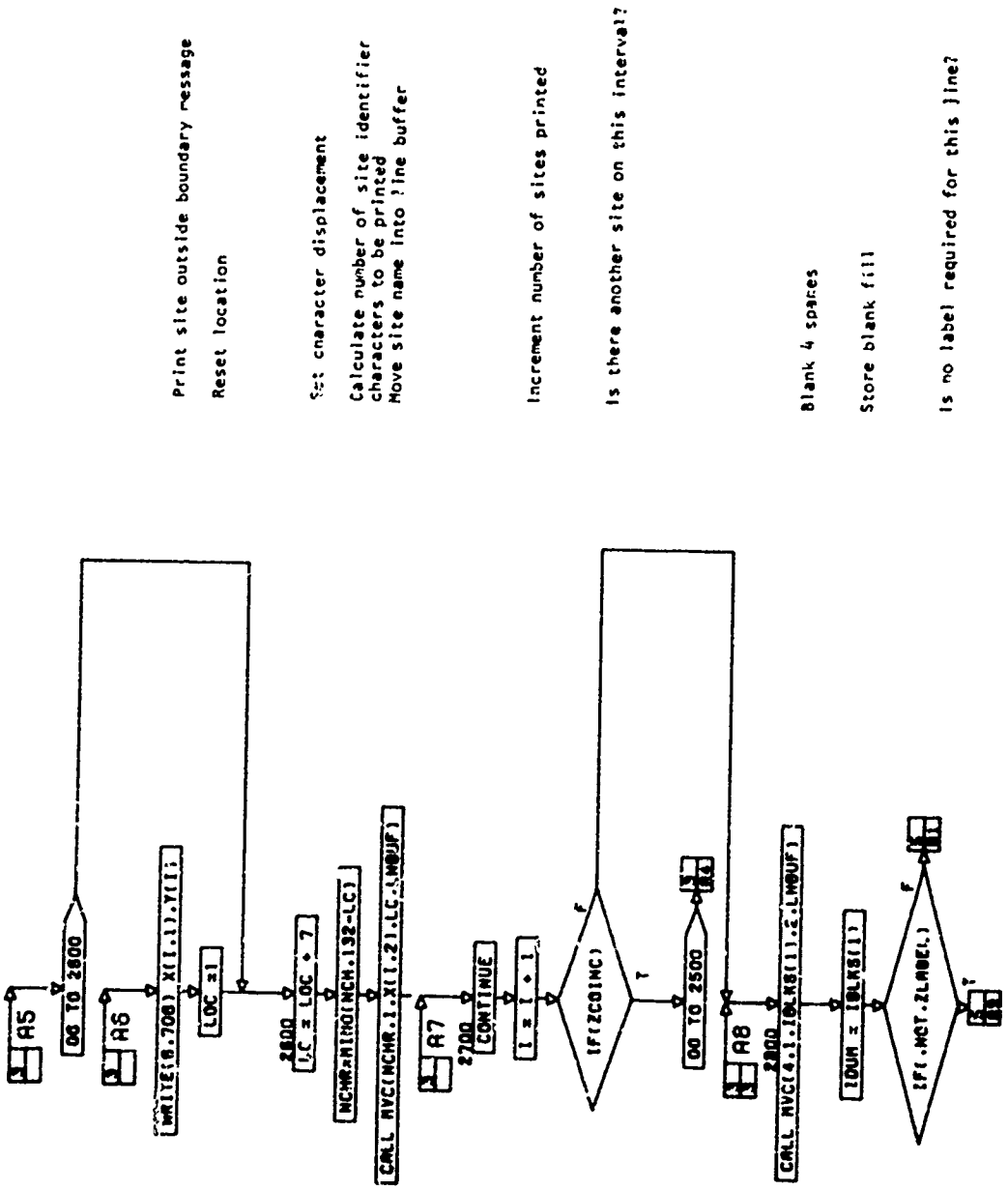
Print map header data

Skip to top of new page

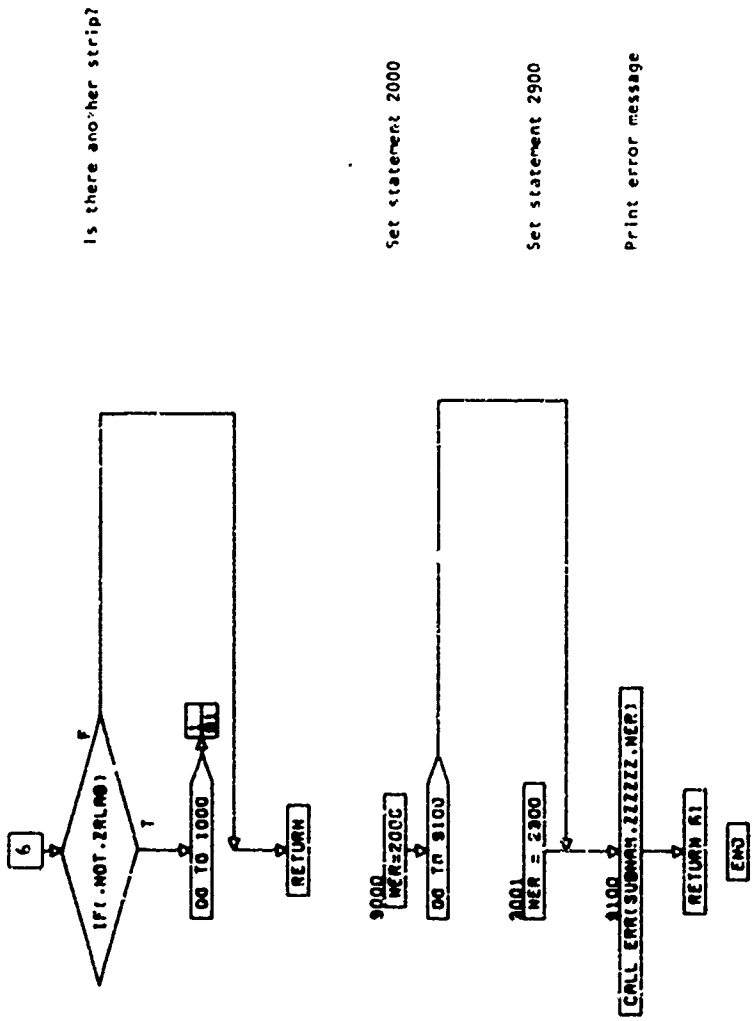

Print abscissa labels for this strip
Initialize first line to northern boundary
Set line interval northern edge
Set line interval southern edge
DO over line buffer
Set label flag
Set initial character location
DO over abscissa axes crossings
Place '.' in line buffer
Increment character location counter
Is no ordinate grid label required for this line?
Move in initial grid characters
DO over line buffer
2
A2
2020 WRITE(6,602) IORBUF
I = 1
YAXIS = YINIT
KO=MLP1
A3
2100 P2 = YAXIS + DY2
P1 = YAXIS - DY2
DO 2150 K=1,14
2150 LNBUF(K)=IBLNKS
ZLABEL = MOD(KO,MLP1).EQ.0
LC = 7
DO 2300 KI=1,MO6
CALL MVC(1,1,ZDOT,LC,ZLNBUF)
2300 LC = LC + 10
IF(.NOT.ZLABEL)
F
T
DO TO 2500
A1
CALL MVC(3,1,ZDOT,0,ZLNBUF)
DO 2400 KI =2,NOW
A3
3

P.4.MAP-2


Move grid into line buffer
Have all the sites been plotted?
Set site out of interval flag
Set next site on interval flag
Does this site lie outside of this interval?
Does this site lie off of this band?
Calculate space location
Does this location lie on this map?
3
2400
LWBUF(IRI) = IDOTS(I)
A4
A4
2500
IF(I.GT.NY)
T
F
GO TO 2800
ZOUTRO = Y(I).LE.P1.OR.Y(I).GT.P2
ZCOINC = Y(I+1).GT.P1.AND.(I+1).LE.P2
IF(ZOUTRO)
T
F
GO TO 2800
F(XI.I).LT.WSTEDO.OR.XI.I).GT.ESTEDO)
T
F
GO TO 2700
LOC = .5 + ((XI.I)-WSTEDO)/DELTAX)
IF(LOC.GE.0.OR.LOC.LE.128)
T
F


Print site outside boundary message
Reset location
Set character displacement
Calculate number of site identifier characters to be printed
Move site name into line buffer
Increment number of sites printed
Is there another site on this interval?
Blank 4 spaces
Store blank fill
Is no label required for this line?
A5
GO TO 2800
A6
WRITE(6,700) XI(1,1),YI(1)
LOC =1
2800
LC = LOC + 7
NCHR=MINO(NCH,132-LC)
CALL MVC(NCHR,1,XI(1,2),LC,LMOUF)
A7
2700
CONTINUE
I = I + 1
IF(I2COINC)
F
T
GO TO 2500
A8
2800
CALL MVC(4,1,IBLKB(1),2,LMOUF)
IOUM = IBLKB(1)
IF(.NOT.ZLABEL)
F
T

Calculate ordinate axis label
Move label into line buffer
Is there another strin?
Store blank fill
Print non line
Set next ordinate line printing increment line counter
Is there more of the text left to be printed?
Print abscissa labels
A9
B1
GO TO 3000
CALL LABELI(IOUN,YAXIS,YORG,HGTA),RETURN(9001))
CALL AVC(4,1,IOUP,2,LNBUF)
IF(.NOT.ZALAB)
GO TO 3000
2800
CALL AVC(4,1,IOUN,1NUX+8,LNBUF)
3000
WRITE(8,502) LNBUF
YAXIS = YAXIS - DELTAY
NO = NO + 1
IF(YAXIS.GE.YMIN)
GO TO 2100
WRITE(8,502) LNBUF
6
T
F


Is there another strip?
Set statement 2000
Set statement 2900
Print error message
6
IF(.NOT.ZALM0)
F
T
GO TO 1000
RETURN
9000
WER=2000
GO TO 9100
2001
WER = 2900
9100
CALL ERRISUBMAN.ZZZZZZ.WER)
RETURN 61
END

1. NAME: MAXN (other entry points: MINN)
2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL
3. LANGUAGE: FORTRAN EXTENDED
4. PURPOSE: Determine the array entry with the greatest (smallest) value.
5. ASSUMPTIONS AND LIMITATIONS: None
6. ERROR RETURNS: None
7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL MAXN (X, NX, RSLT)

   entry point CALL MINN (X, NX, RLST)

   X - Array from which the maximum(minimum)entry is to be found.

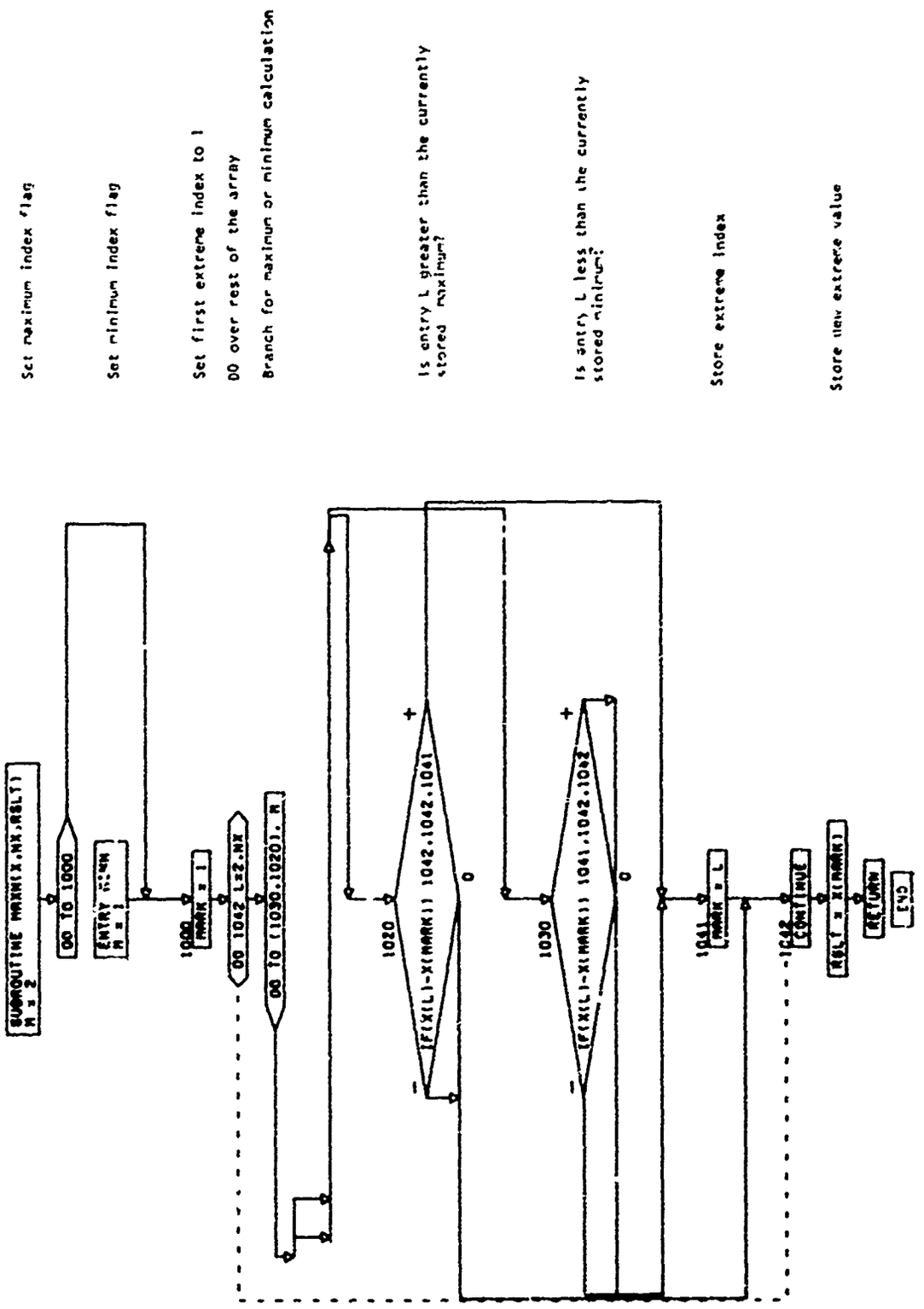
   NX - Integer, number of entries from array X to be considered.

   RSLT - Maximum (minimum) entry from array X.
8. PROGRAMS CALLING THIS PROGRAM:

   MAXN: SØRTR

   entry point MINN: SØRTR
9. COMMON VARIABLES USED: None
10. COMMON VARIABLES TO BE SET: None
11. COMMON VARIABLES CHANGED: None
12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None
13. NOTES ON METHODOLOGY: If entry is via MAXN, the mode index (M) is set to 2 indicating that the maximum value of the array is desired. Entry via MINN sets M to 1 indicating that the minimum value is desired from the array. At statement 1000 the array index (MARK) of the maximum (or minimum) is initialized to one. A DO loop which ranges over all the entries (NX) considered from the array X is entered. If M is one, statement

1030 is used to determine if the array entry being tested (X(L)) is less than the currently stored minimum, X(MARK). If X(L) is less than X(MARK), control transfers to statement 1041 where the array index (MARK) for the minimum array entry is set to the present entry index (L). If M is two, statement 1020 is used to determine if the array entry being tested (X(L)) is greater than the currently stored maximum X(MARK). If X(L) is greater than X(MARK), control transfers to statement 1041 where the array index (MARK) for the maximum array entry is set to the present entry index (L). Upon completion of the above loop the maximum (M=2) or minimum (M=1) value for the array is stored in RSLT and control returned to the calling program.


SUBROUTINE MAXN(X,NX,RSLT)
M = 2
DO TO 1000
ENTRY MINN
M = 1
1000 MARK = 1
DO 1042 L=2,NX
DO TO (1030,1020), M
1020
IF(X(L)-X(MARK)) 1042,1042,1041
0
-
1030
IF(X(L)-X(MARK)) 1041,1042,1042
0
-
1041 MARK = L
1042 CONTINUE
RSLT = X(MARK)
RETURN
END
Set maximum index flag
Set minimum index flag
Set first extreme index to 1
DO over rest of the array
Branch for maximum or minimum calculation
Is entry L greater than the currently stored maximum?
Is entry L less than the currently stored minimum?
Store extreme index
Store new extreme value

P.4.MAXN-1

1. NAME: MVC

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Manipulate character strings to form UTM grid square designators.

5. ASSUMPTIONS AND LIMITATIONS: If the arguments IØPTR and INPTR are assigned values less than zero, the characters will not be moved.

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL MVC (ICNT, INPTR, ZINADD, IØPTR, ZIØADD)

   ICNT - Integer, number of characters to be moved from ZINADD to ZIØADD.

   INPTR - Integer, index of the first character to be moved from ZINADD.

   ZINADD - Character, 2 word array (20 character) from which characters will be moved.

   IØPTR - Integer, index of the location in ZIØADD where the first character will be moved.

   ZIØADD - Character, 2 word array (20 characters) where the characters will be moved.

8. PROGRAMS CALLING THIS PROGRAM: ABSCØR, CLN, CØØRT, LABEL, MAIN, MAP

9. COMMON VARIABLES USED: None

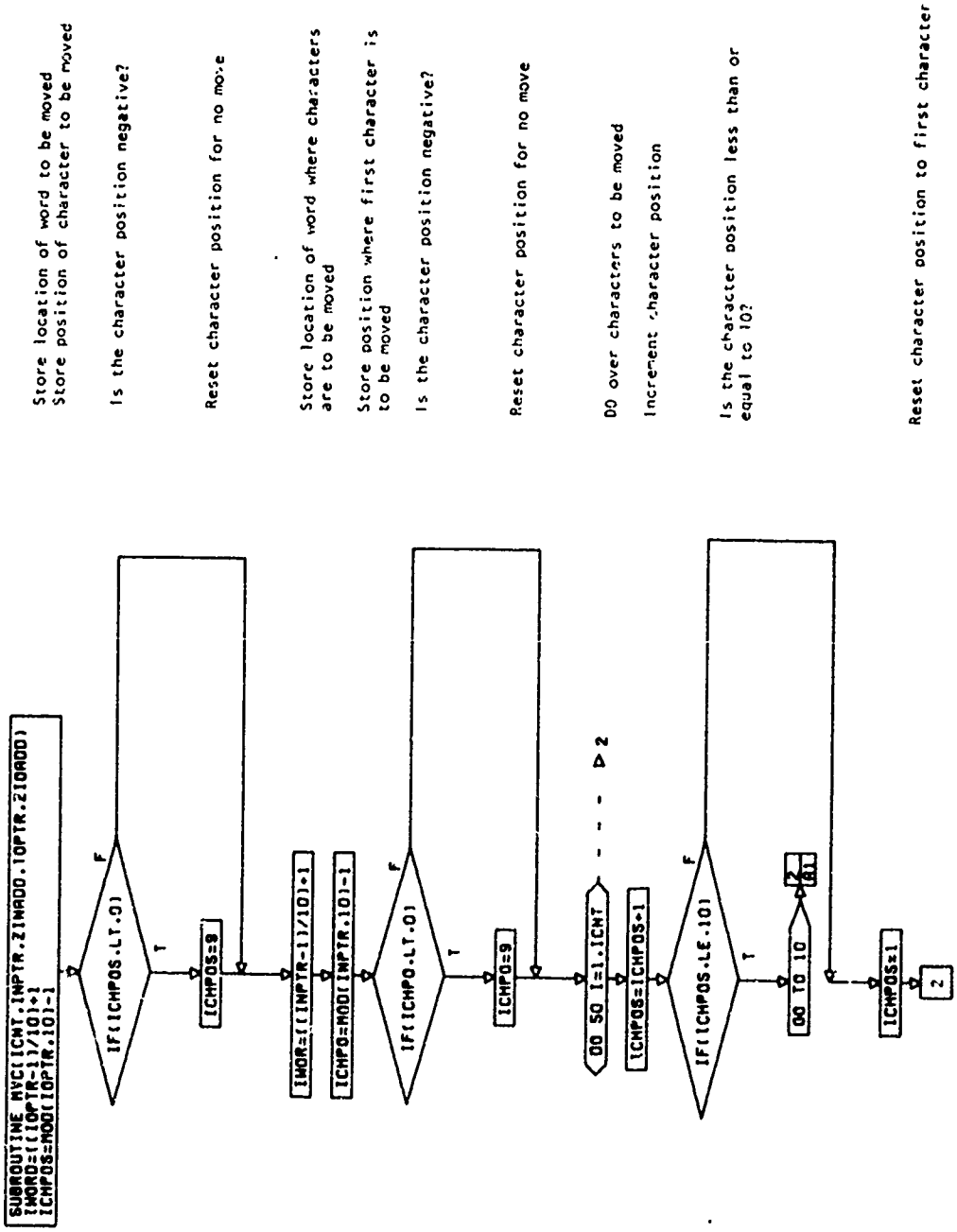
10. COMMON VARIABLES TO BE SET: None

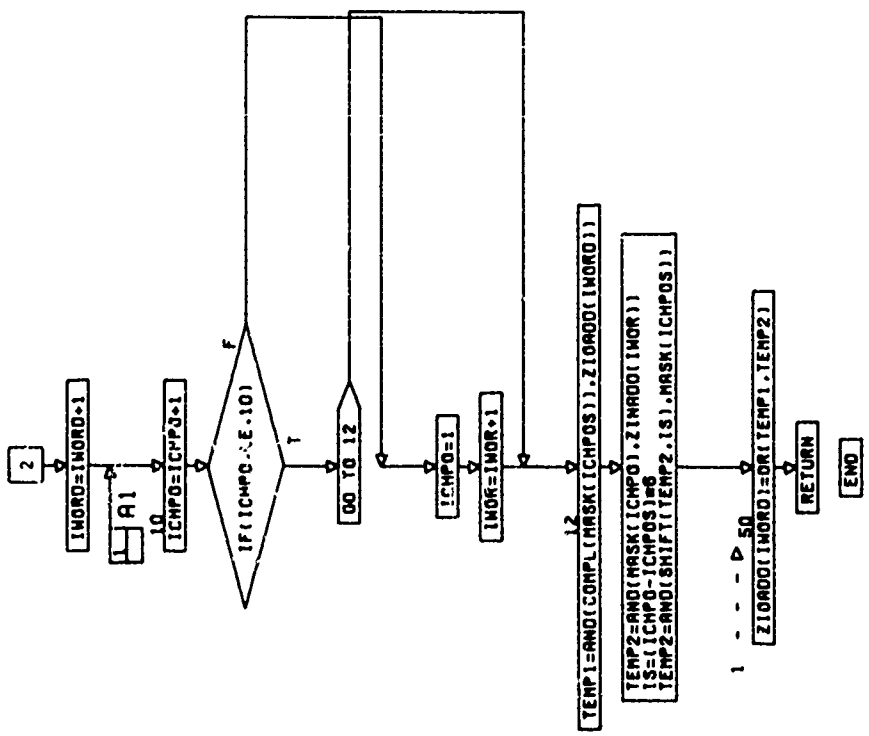
11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF ARGUMENTS: None

13. NOTES ON METHODOLOGY: Upon entry to subroutine MVC, the word location (IWØRD) in the array ZIØADD where the characters are to be moved and the index (ICHPØS) of the first character position within that word to be filled are calculated. If ICHPØS is negative, ICHPØS is set to 9. Next, the location (IWØD) of the word in the array ZINADD from which the character string is to be moved and the index (ICHPØ) of the first character position to be moved from that word are calculated. If ICHPØ is negative, ICHPØ is reset to 9. Next, a DO loop is entered which ranges over the number of characters to be moved (ICNT).

Upon entry to this loop, the position (ICHPØS) in word ZIØADD (WØRD) where the first character will be moved is found by adding one to the character index ICHPØS. If ICHPØS lies within word ZIØADD (WØRD) (ICHPØS less than or equal 10) control transfers to statement 10. If ICHPØS is greater than 10, the character position is reset to one and the word location IWØRD incremented by one. At statement 10 the position (ICHØ) of the first character to be moved from the word ZINADD (IWØR) is found by incrementing ICHØ by one. ICHPØ is then tested. If ICHPØ is greater than 10, the character position lies in the following word and ICHPØ is reset to one and the word location incremented by one. At statement 12, TEMP1 is set to the character string ZIØADD (IWØRD) with the ICHPØS character position set to 00 octal. TEMP2 is set to a field of blanks with the ICHPØ character position set to the ICHPØ character position of ZINADD (IWØR). TEMP2 is then shifted by the difference in position of the two characters and masked by an octal 7 in character position ICHPØS. This result is stored in TEMP2 ZIØADD (IWØRD) is then the ØR mask of TEMP1 and TEMP2. The loop increments ICHPØS and ICHPØ until ICNT characters have been moved and control returns to the calling program.


SUBROUTINE MVC(ICNT,INPTR,ZIWADD,IOPTR,ZIOADD)
IWORD=((IOPTR-1)/10)+1
ICHPOS=MOD((IOPTR,10)-1
Store location of word to be moved
Store position of character to be moved
Is the character position negative?
IF((ICHPOS.LT.0)
F
T
Reset character position for no move
ICHPOS=8
Store location of word where characters are to be moved
IWOR=((INPTR-1)/10)+1
Store position where first character is to be moved
ICHPO=MOD((INPTR,10)-1
Is the character position negative?
IF((ICHPO.LT.0)
Reset character position for no move
ICHPO=9
DO over characters to be moved
DO 50 I=1,ICNT
Increment character position
ICHPOS=ICHPOS+1
Is the character position less than or equal to 10?
IF((ICHPOS.LE.10)
GO TO 10
Reset character position to first character
ICHPOS=1
2
50
2
1


Increment location to next word
.ncrement character position
Is the character position less than or equal to 10?
Reset character position to first character
Increment location to next word
Mask and shift character strings
Calculate new character string
2
IWORD=IWORD+1
A1
10
ICHPO=ICHPO+1
IF(ICHPO.LE.10)
F
T
GO TO 12
ICHPO=1
IWOR=IWOR+1
12
TEMP1=AND(COMPL(MASK(ICHPOS)),ZIOADD(IWORD))
TEMP2=AND(MASK(ICHPO),ZINADD(IWOR))
IS=(ICHPO-ICHPOS)*8
TEMP2=AND(SHIFT(TEMP2,IS),MASK(ICHPOS))
1
50
ZIOADD(IWORD)=OR(TEMP1,TEMP2)
RETURN
END

1. NAME: MYTIME
2. TYPE OF PROGRAM: FUNCTION, MATHEMATICAL
3. LANGUAGE USED: FORTRAN EXTENDED
4. PURPOSE: Calculate accumulated CP time in units of 100ths of a second
5. ASSUMPTIONS AND LIMITATIONS: None
6. ERROR RETURNS: None
7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   I = MYTIME (I)

   I - Integer, accumulated CP time (100ths seconds).
8. PROGRAMS CALLING THIS PROGRAM: JTIMER, PRODMF
9. COMMON VARIABLES USED: None
10. COMMON VARIABLES SET: None
11. COMMON VARIABLES CHANGED: None
12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None
13. NOTES ON METHODOLOGY: Function MYTIME calculates the accumulated Control Processor (CP) time in units of 100ths of a second. Upon entry MYTIME uses the function SECOND from the FORTRAN EXTENDED library to find the accumulated CP time in second. MYTIME is calculated by multiplying this time by 100 to get accumulated time in 100ths of a second. Control is then returned to the calling routine.

Calculate time in 100ths seconds

```
FUNCTION  MYTIME(JDUM)
DTCP = SECOND(0)
MYTIME = DTCP*100.0
```

RETURN

END

1. NAME: ØNEPT

2. TYPE OF PROGRAM: SUBROUTINE, MATHEMATICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Calculate the terrain elevation at a prespecified grid point.

5. ASSUMPTIONS AND LIMITATIONS: The terrain data is accessible from the direct access file and the terrain directory has been previously stored.

6. ERROR RETURNS: A RETURN A1 will be executed if either of the following errors are detected:

   A. The prespecified point coordinates cannot be converted to UTM.

   B. The 2 letter UTM grid square designator for the square of interest cannot be converted to relative easting and northing.

   C. The grid square containing the point under consideration is not on the direct access terrain file.

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL ØNEPT (XS, YS, H), RETURNS (A1)

   XS - Relative easting of the grid point under consideration.

   YS - Relative northing of the grid point under consideration.

   H - MSL altitude of the terrain at the grid point under consideration.

   A1 - Exit which is taken if one of the errors described in 6 (above) occurs.

8. PROGRAM CALLING THIS PROGRAM: ELEV

9. COMMON VARIABLES USED:

   A. ZZBUF - IBUF

   B. ZZFASC - LØDA, LTDA, NØDA

C. ZZCV2 - LTRS, NBRS

10. COMMON VARIABLES TO BE SET:

A. ZZBUF - IBUF

B. ZZFASC - LØDA, LTDA, NØDA

C. ZZCV2 - LTRS

11. COMMON VARIABLES CHANGED:

A. ZZCV2 - NBRS

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

A. SUBROUTINES

(2) CALL CITA (X, Y, L), RETURNS (A1) (entry point to CØØRT)- Converts the relative easting and northing of the point under consideration to UTM coordinates (2 letters, 8 digits).

X - Relative easting of the point under consideration.

Y - Relative northing of the point under consideration.

L - Dummy variable not used for this entry point.

A1 - Control is transferred to this statement number if an error is detected in CITA.

(3) CALL CITN (X, Y, L), RETURNS (A1) (entry point to CØØRT) - Converts the UTM coordinates of a point to relative easting and northing.

X - Relative easting of the point under consideration.

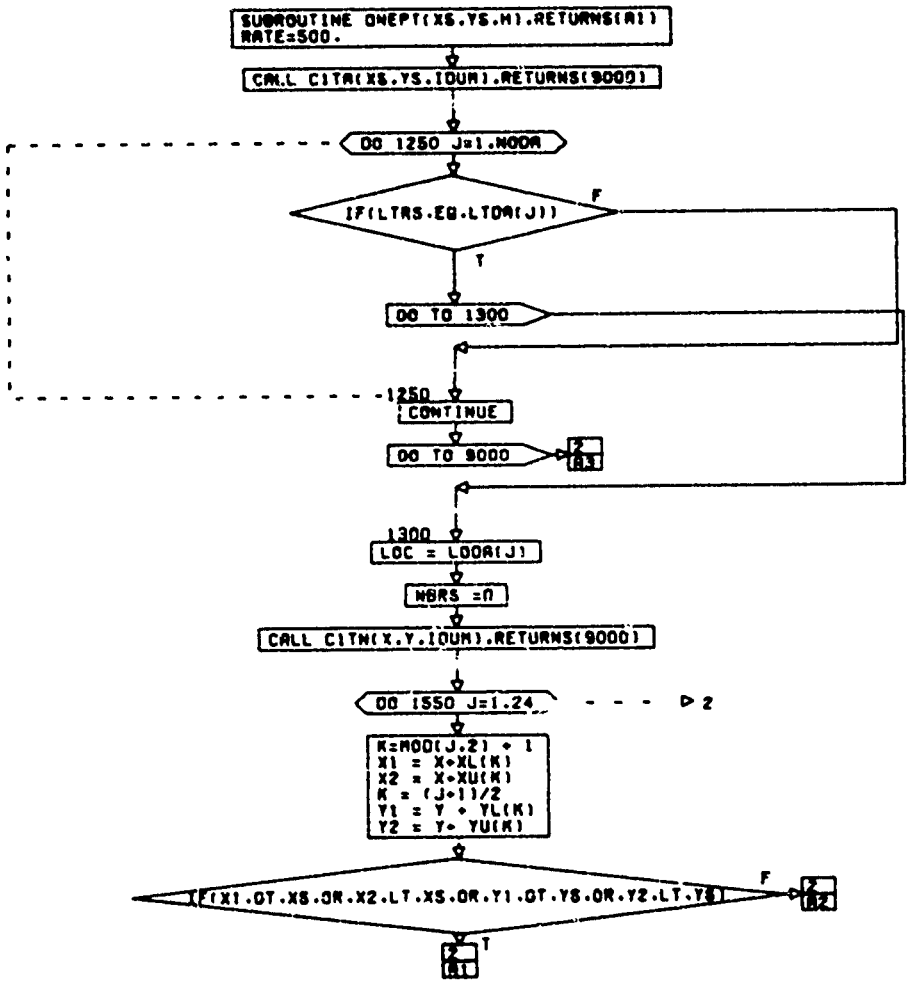
Y - Relative northing of the point under consideration.

L - Dummy variable not used.

A1 - Control is transferred to this statement number if an error is detected in CITN.

13. NOTES ON METHODOLOGY: Upon entry to ØNEPT, the lateral data rate, RATE, is set to 500 meters. CITA is then called to calculate the UTM coordinates of the location (XS,YS). A DO loop is then used to determine if the grid square (LTRS) which contains point (XS, YS) is available from the direct access terrain file. If this grid square is not on the terrain file, control transfers to statement 9000 where a RETURN Al is executed. If the grid square is available from the direct access file, control transfers to statement 1300 where the address of the first terrain record for this square is stored in local variable LØC. CITN is then called to calculate the relative easting and northing of the southwest corner of this grid square.

A DO loop is then entered. This loop ranges over the 24 records containing the terrain elevations for this 100 km grid square. Each record covers a rectangular area where the sides are defined by the boundaries X1, (western), X2 (eastern), Y1 (southern) and Y2 (northern). A test is then performed to determine if the point (XS, YS) is in this rectangle. If the point is not within this record area, control transfers to the end of the loop. The record number is incremented by one and the next record area is tested. When the record which covers (XS, YS) has been found, then the point number within the record and the record address IR is not equal to LASLØC (1), the required record is read from the direct access file and stored in IBUF. The MSL altitude is then calculated by multiplying the altitude IBUF (IB) by the vertical data rate (10 meters). Control then returns to the calling program.



Set lateral data rate

Convert point relative coordinates to UTM

DO over available grid squares

Is the grid square containing the point of interest available?

Store address of first record for this square

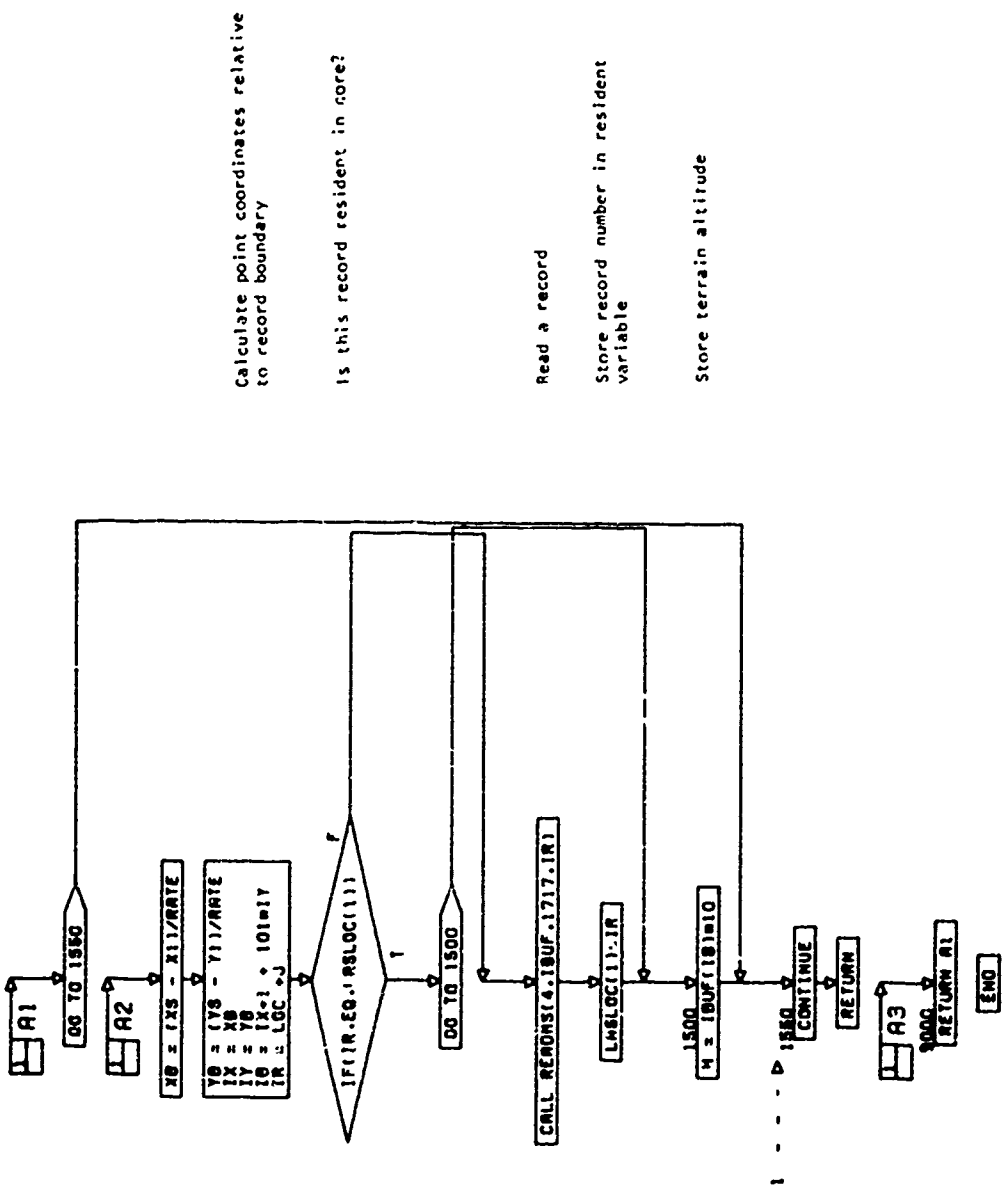
Convert point relative coordinates to UTM

DO over records for this grid square

Calculate record boundaries

Does the point lie within this record's boundary?

P.4.ØNEPT-1


Calculate point coordinates relative to record boundary
Is this record resident in core?
Read a record
Store record number in resident variable
Store terrain altitude
A1
GO TO 1550
A2
XB = (XS - X11)/RATE
YB = (YS - Y11)/RATE
IX = XB
IY = YB
IB = IX-1 * 101+IY
IR = LOC *J
IF(IR.EQ.LRSLOC(1)) T F
GO TO 1500
CALL READRS(4,IBUF,1717,IR)
LRSLOC(1)=IR
1500
M = IBUF(IB)+10
1550 CONTINUE
RETURN
A3
9000
RETURN A1
END

1. NAME: PACK

2. TYPE OF PROGRAM: SUBROUTINE, OUTPUT

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Punch a track identifier, track point identifier and logical flag on file 9.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:
   Call PACK (I, ISITE, ZWENG)
   I - Integer, track index.
   ISITE - Integer, site index.
   ZWENG - Logical, site-track engageability flag.

8. PROGRAMS CALLING THIS PROGRAM: MAIN

9. COMMON VARIABLES USED: None

10. COMMON VARIABLES TO BE SET: None

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None

13. NOTES ON METHODOLOGY: Each time PACK is called, a record is punched on file 9 containing the track identifier, the track point identifier and the engagement flag. Control is then returned to the calling program.

Punch targeted track identifier on file 9

SUBROUTINE PACK (I,ISITE,ZNEW1)
WRITE(9,500)ITRK(I),ISITE,ZNEW

RETURN

END

1. NAME: PMAP

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Control the input of the terrain data from tape and initialize program execution.

5. ASSUMPTIONS AND LIMITATIONS: File 7 contains the terrain data tape.

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS: The linkage to PMAP is from the SCØPE operating system.

8. PROGRAMS CALLING THIS PROGRAM: None

9. COMMON VARIABLES USED:
   A. ZZRAI - IMAX
   B. ZZTRKD - NDWØRD

10. COMMON VARIABLES TO BE SET: None

11. COMMON VARIABLES CHANGED:
   A. ZZRAI - IMAX
   B. ZZTRUD - NDWØRD

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:
   A. SUBROUTINES
      (1) CALL MAIN, RETURNS (A1) - Controls input of data cards, checking of input data and plotting of maps.

          A1 - Control is transferred to this statement number if an error is detected in MAIN.

(2) CALL RAW (TIN, TØUT), RETURNS (Ai) - Read terrain data from a magnetic tape (file TIN) and write this data on the direct access file TØUT.

TIN - Integer, file number of tape to be read.

TØUT - Integer, file number of direct access device where the terrain data will be stored.

Ai - Control is transferred to this statement number if an error is detected in RAW.

13. NOTES ON METHODOLOGY: Upon entry to PMAP, NDWØRD is set to 3000, KINN is set to 7 and KØUT is set to 4. SKIP and NSQUAR are then read from the terrain tape (file 7). The maximum number of terrain records (IMAX) to be read from the terrain file is then calculated. RAW is then called to read the terrain data from the tape (file 7) and store this terrain data on the direct access device (file 4). MAIN is then called to provide the logical sequencing required for successful PMAP execution. Control then returns to the SCØPE operating system.

PROGRAM PMAP(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8,TAPE9,
TAPE4,TAPE7)
NOWORD=3000
KOUT=4
KINN=7
READ(KINN)SKIP,NSQUAR
LMAX=24*NSQUAR + 2
CALL RAWIO(KINN,KOUT)
CALL MAIN,RETURNS(100)
100
CONTINUE
STOP
END
Store input and direct access terrain
file number
Read limits from terrain tape
Set maximum numbers of records to be
transferred
Transfer terrain file from tape to
direct access device
Perform logical PMAP functions

1. NAME: RAW

2. TYPE OF PROGRAM: SUBROUTINE, INPUT/OUTPUT

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Read the terrain data file from type, create a directory, and write the directory and terrain data on a direct access storage file.

5. ASSUMPTIONS AND LIMITATIONS: The tape (file 10) has the directory and terrain data in the proper sequence. The direct access storage file (file 11) must be made available.

6. ERROR RETURNS: A RETURN A1 will occur if the following error is detected:

   A. The number of records read exceeds the maximum IMAX.

7. LINKAGE STATEMENTS AND DESCRIPTION OF ARGUMENTS:

   CALL RAW (TIN, TOUT), RETURN (A1)

   TIN - Number of the input tape file (10).

   TOUT - Number of the output direct access file (11).

   A1 - Exit taken if the condition listed under 6 (above) occurs.

8. PROGRAMS CALLING THIS PROGRAM: PMAP

9. COMMON VARIABLES USED:

   A. ZZBUF - IBUF

   B. ZZRA1 - IMAX, IRAN

10. COMMON VARIABLES TO BE SET:

   A. ZZRA1 - IMAX

11. COMMON VARIABLES CHANGED:

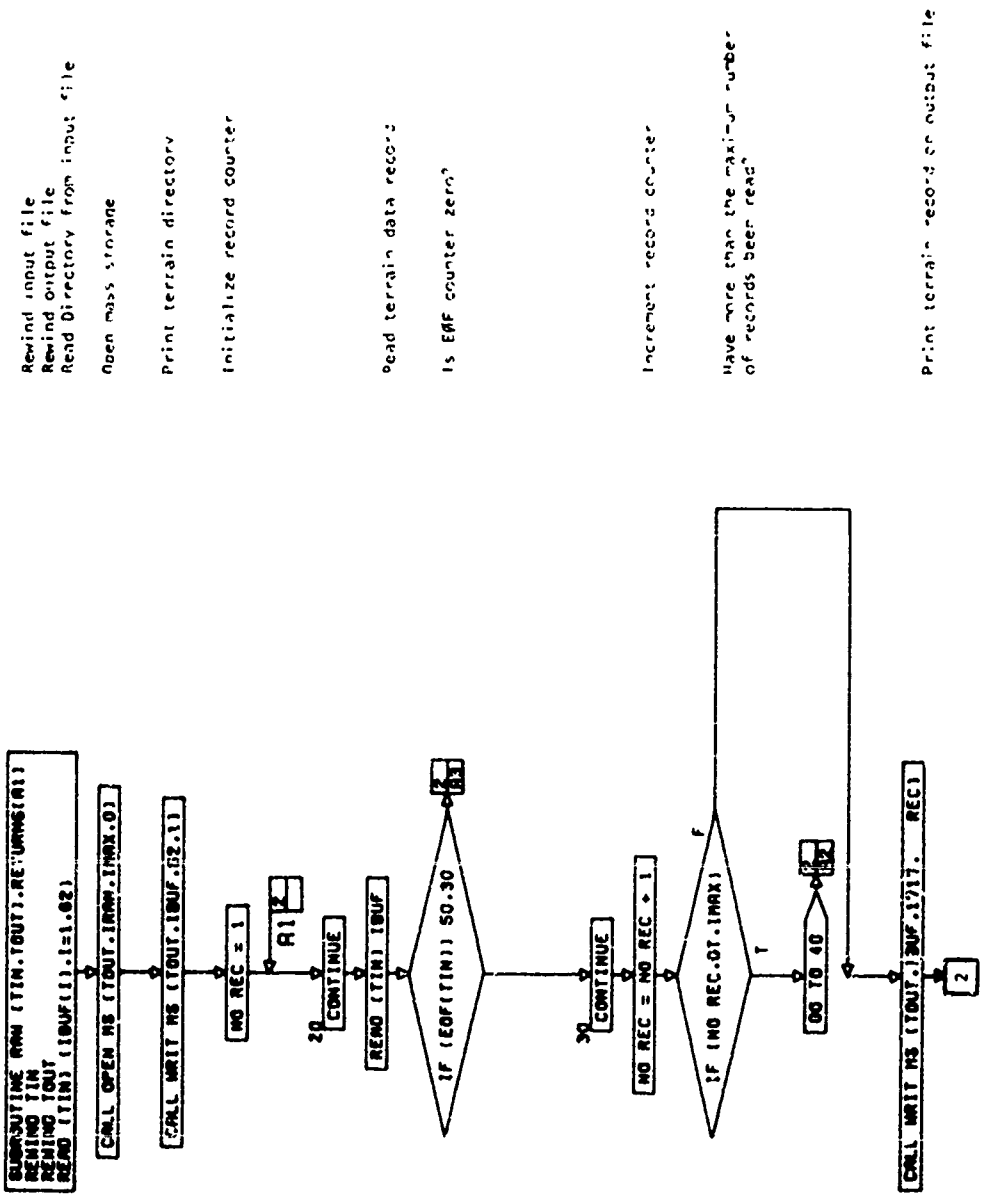
   A. ZZBUF - IBUF

   B. ZZRA1 - IRAN

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None.

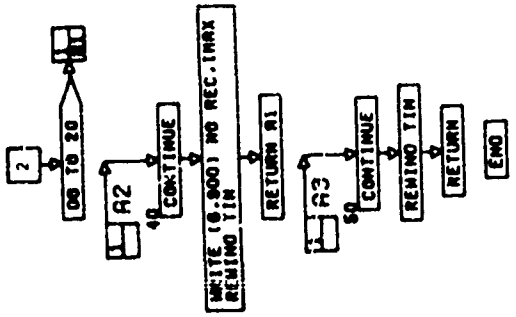
13. NOTES ON METHODOLOGY: RAW reads the terrain data from tape (file TIN) and writes these data on the direct access device (file TØUT). Upon entry to RAW a REWIND is executed for both files to provide the proper starting reference for these files. Sixty-two words containing the grid zone identifier, grid square identifiers and terrain data locations are read from tape (file TIN), stored in IBUF, and then written on the direct access device (file TØUT).

Next the record counter (NØ REC) is set to one and a loop entered. This loop increments the record counter, reads a record (1717 or 1515 terrain data points) from tape and writes this data on the direct access device (file TØUT). This loop reads records from the tape and writes them on the direct access device until an end of file mark is encountered or the maximum number of records (IMAX) is exceeded. If IMAX is exceeded, control is transferred to statement 40 where an error message is printed, a REWIND TIN is executed, and an error exit (RETURN A1) taken. If an end of file is encountered, control transfers to statement 80 where a REWIND TIN is executed and a normal return taken.


SUBROUTINE RAW (TIN,TOUT),RETURNS(A1)
REWIND TIN
REWIND TOUT
READ (TIN) (IBUF(I),I=1,62)
CALL OPEN MS (TOUT,IRAW,IMAX,0)
CALL WRIT MS (TOUT,IBUF,62,1)
NO REC = 1
A1
20 CONTINUE
READ (TIN) IBUF
IF (EOF(TIN)) 50,30
30 CONTINUE
NO REC = NO REC + 1
IF (NO REC.GT.IMAX)
T
GO TO 40
F
CALL WRIT MS (TOUT,IBUF,1717, REC)
Rewind input file
Rewind output file
Read Directory from input file
Open mass storage
Print terrain directory
Initialize record counter
Read terrain data record
Is EOF counter zero?
Increment record counter
Have more than the maximum number of records been read?
Print terrain record on output file

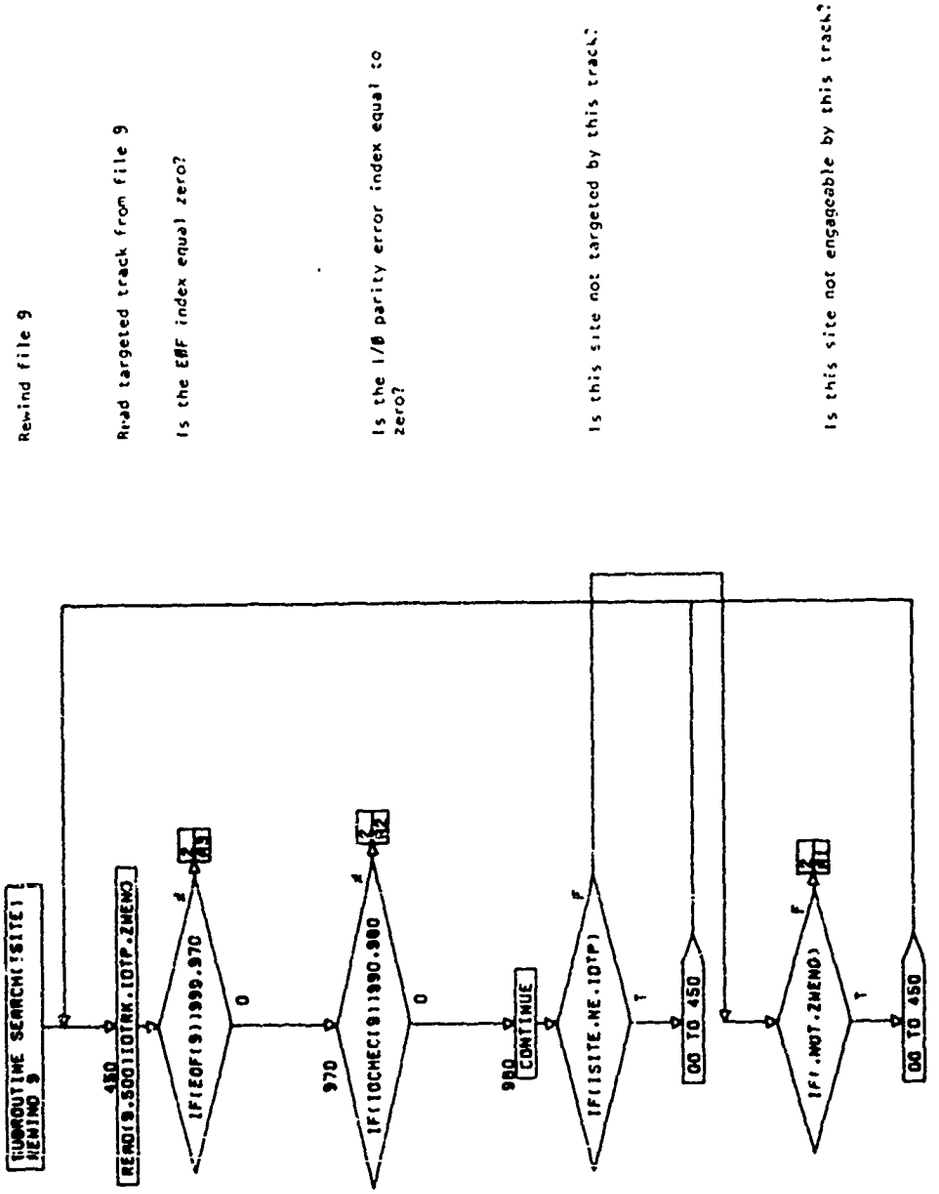
Print overflow message

Rewind input file



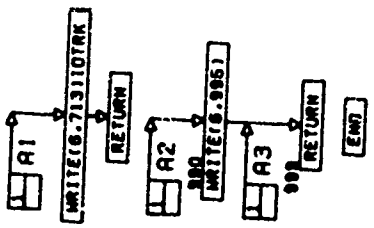
1. NAME: SEARCH
2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL
3. LANGUAGE: FORTRAN EXTENDED
4. PURPOSE: Find the targeted track point for a specified site.
5. ASSUMPTIONS AND LIMITATIONS: None
6. ERROR RETURNS: None
7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:
   CALL SEARCH (ISITE)
   ISITE - Integer, specified site index.
8. PROGRAMS CALLING THIS PROGRAM: MAIN
9. COMMON VARIABLES USED: None
10. COMMON VARIABLES TO BE SET: None
11. COMMON VARIABLES CHANGED: None
12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None
13. NOTES ON METHODOLOGY: Upon entry to subroutine SEARCH, a REWIND is executed on file 9 to assure that this file starts at the proper position. The first record is read from file 9 at statement 450. If an end file is encountered, control transfers to statement 999 where a normal return is taken to the calling program. At statement 970, a check is performed to determine if an 1/Ø parity error was detected during this read. If an 1/Ø parity error was detected, control transfers to statement 990 where an error message is printed and control returns to the calling program. If an end of file is not encountered and an 1/Ø parity error is not detected, the track point identifier (IDTP) read from this record is tested against the site identifier (ISITE) specified in the argument list. If these identifiers are not equal, control

transfers to statement 450 where the next record will be read. If these identifiers are equal, the engagement flag ZWENC is tested. If the engagement flag is TRUE, a targeted-by-track message is printed and control returns to the calling subroutine. Otherwise, control returns to statement 450 where the next record will be read.


SUBROUTINE SEARCH(!SITE)
REWIND 9
450
READ(9,500)IOTRK,IOTP,ZWENO
IF(EOF(9))999,970
0
970
IF(IOCHEC(9))990,980
0
980
CONTINUE
IF(ISITE.NE.IOTP)
T
GO TO 450
IF(.NOT.ZWENO)
T
GO TO 450
Rewind file 9
Read targeted track from file 9
Is the EOF index equal zero?
Is the I/O parity error index equal to zero?
Is this site not targeted by this track?
Is this site not engageable by this track?

Print targeted by track identification message

Print error message



1. NAME: SETBPT

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Calculate the block pointer address for a specified track and point.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:
   CALL SETBPT (NTK, I, IWØRD)
   NTK - Index for the desired track.
   I - Index of the desired end point.
   IWØRD - Block pointer address for desired point.

8. PROGRAMS CALLING THIS PROGRAM: JTRK, TTRK, ZCKPTI

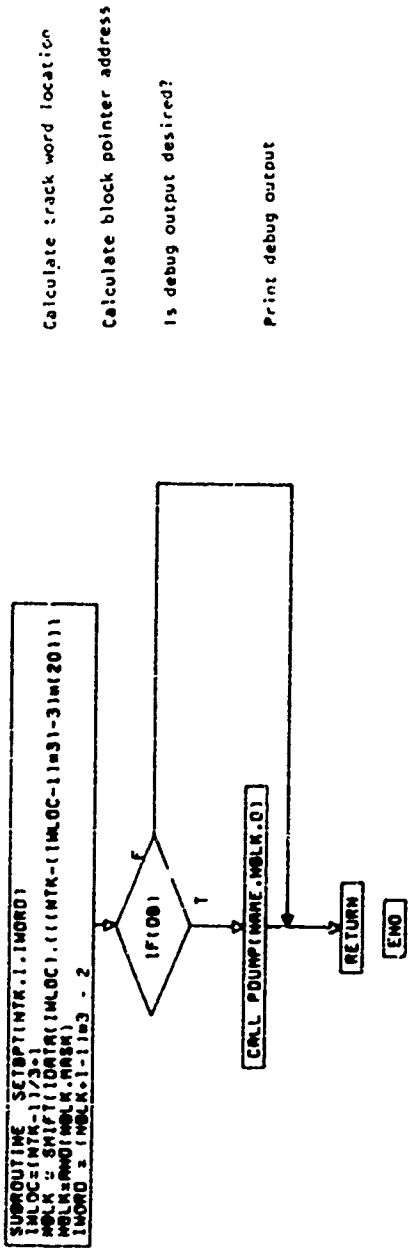
9. COMMON VARIABLES USED:
   A. ZZTRKD - IDATA (equivalenced to TKDATA)

10. COMMON VARIABLES TO BE SET:
    A. ZZTRKD - IDATA (equivalenced to TKDATA)

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None

13. NOTES ON METHODOLOGY: Upon entry to SETBPT the location of the word (IWLØC) in the track data array containing this block pointer address is calculated for this track. The block number (NBLK) is then determined via a mask expression and shift of the word found at location IWLØC. The block pointer word (IWØRD) is then determined from NBLK and the point index I. If the debug flag is set to TRUE, the track data array is dumped. Control then returns to the calling program.


SUBROUTINE SETBPT(NTK,I,IWORD)
INLOC=(NTK-1)/3+1
MBLK = SHIFT(IDATA(INLOC),(((NTK-(INLOC-1)*3)-3)*(20)))
MBLK=AND(MBLK,MASK)
IWORD = (MBLK+1-1)*3 - 2
IF(DB)
F
T
CALL POUMP(NAME,MBLK,0)
RETURN
END
Calculate track word location
Calculate block pointer address
Is debug output desired?
Print debug output

1. NAME: SØRTR

2. TYPE OF PROGRAM: SUBROUTINE, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Perform either ascending or descending sort on an array and output maximums and minimums.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL SØRTR (X,Y,NX,MAXNX,NY,MØDE)

   X - Array for which an ascending sort, descending sort or maximum and minimum of the array entries will be calculated.

   Y - Array whose entries will be moved in an order corresponding to the sort performed on X.

   NX - Integer, number of entries from the array X to be available for sort.

   MAXNX - Integer, maximum number of entries upon which a sort is to be performed.

   NY - Integer, number of entries in array Y which will be moved for each corresponding move in the X array.

   MODE - Integer, mode of operation for this routine. 1 implies descending sort; 2 implies ascending sort; 3 implies maximum and minimum entries only are desired.

8. PROGRAM CALLING THIS PROGRAM: MAIN

9. COMMON VARIABLES USED:

   A. ZZMAPP - XMAX, YMAX, XMIN, YMIN

10. COMMON VARIABLES TO BE SET: None

11. COMMON VARIABLES CHANGED:

A. ZZMAPP - XMAX, YMAX, XMIN, YMIN

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

A. SUBROUTINES

(1) CALL MAXN(X,NX,RSLT) - Find the maximum entry in array X.

X - Array from which the maximum entry is to be found.

NX - Integer, number of entries from array X to be considered.

RSLT - Maximum entry from array X.

(2) CALL MINN(X,NX,RSLT) (entry point to MAXN) - Find the the minimum entry in array X.

X - Array from which the maximum entry is to be found.

NX - Integer, number of entries from array X to be considered.

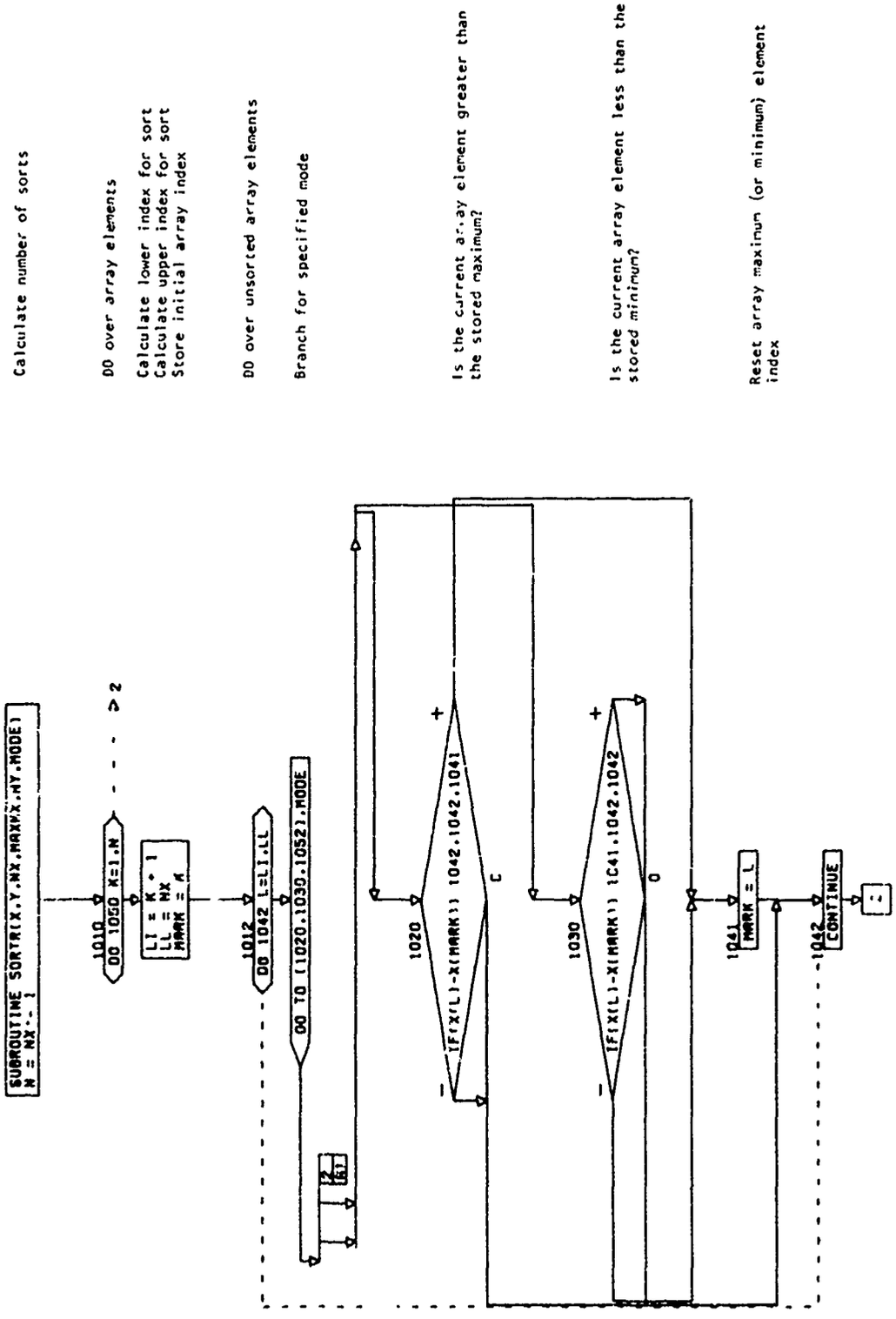
RSLT - Maximum entry from array X.

13. NOTES ON METHODOLOGY: Upon entry to SØRTR, N is set to the number of array entries to be sorted (NX) minus one. At statement 1010 the sort DO loop is entered. First the lower (LI) and upper (LL) indices which determine the entries which require sort are determined. The lower index is set to the DO loop index K and the upper index is set to NX. The array index MARK is set to the DO loop index K. At statement 1012, a DO loop which ranges over the entries (LI to LL) to be sorted from the array X is entered. If MØDE is one, statement 1020 is used to determine if the array entry being tested (X(L)) is greater than the currently stores maximum, X(MARK). If X(L) is greater than X(MARK), control transfers to statement 1041 where the array index (MARK) for the

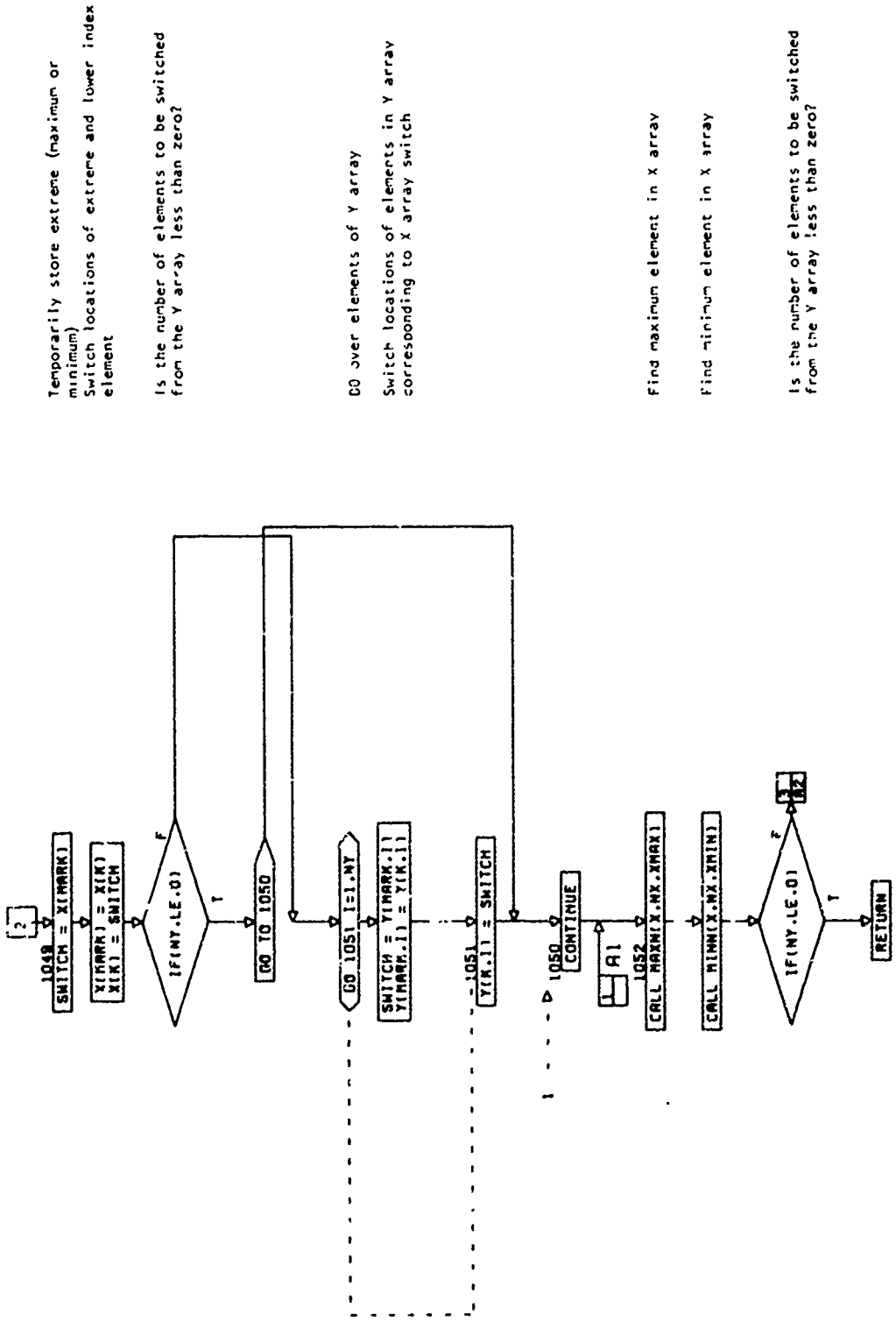
maximum array entry is set to the present entry index (L). If MØDE is two, statement 1030 is used to determine if the array entry being tested (X(L)) is less than the currently stored minimum, X(MARK). If X(L) is less than X(MARK), control transfers to statement 1041 where the array index (MARK) for the minimum array entry is set to the present entry index (L). If MØDE is 3, no sorts are required and control transfers to statement 1052. Following statement 1049 the maximum (MØDE=1) or minimum (MØDE=2) array entry is stored in the array location K and the value of the array previously stored in array location K is stored in location MARK. This procedure stores the values X(MARK) in descending order (MØDE=1) or ascending order (MØDE=2) as required.

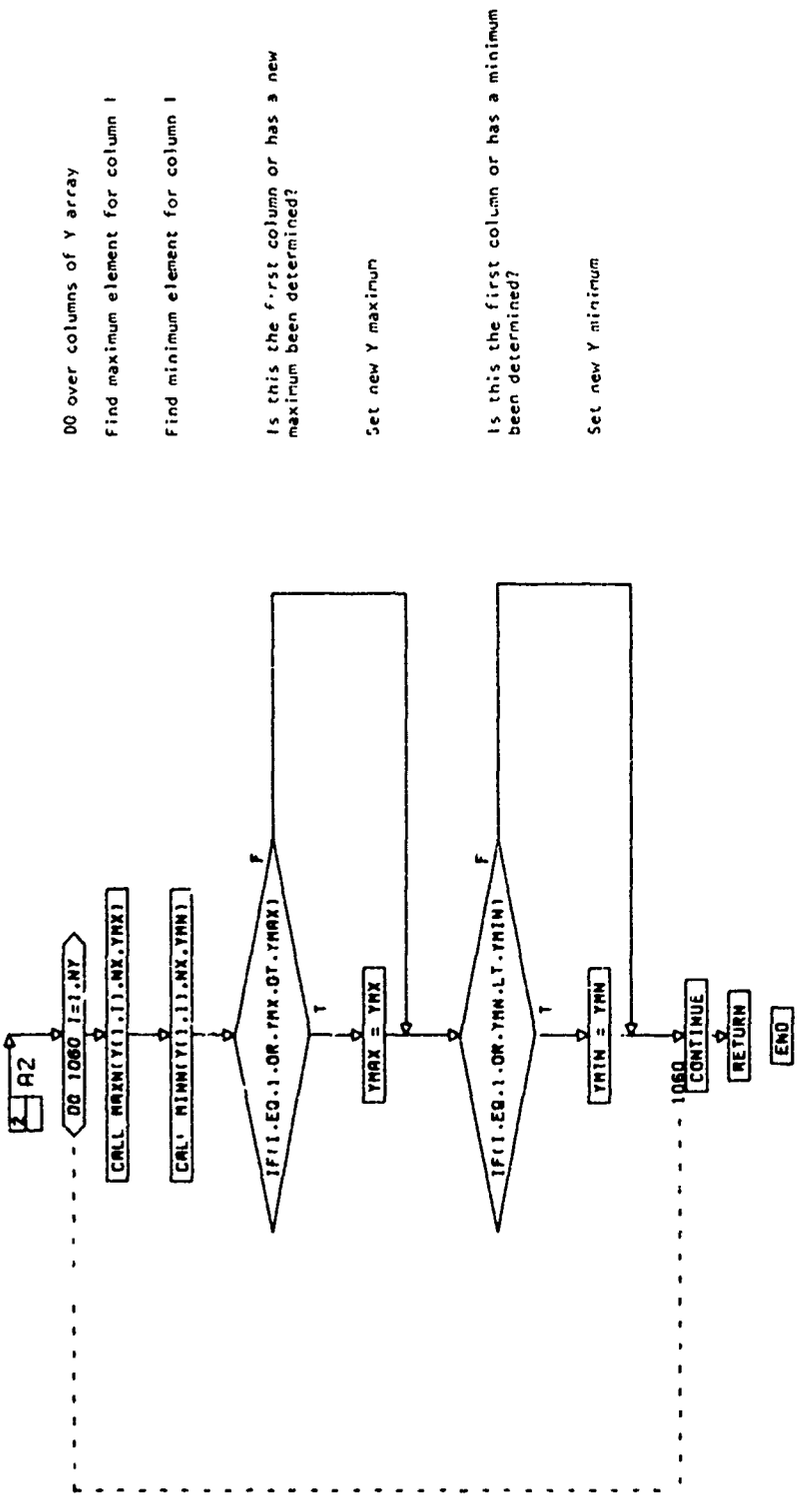
If NY is greater than zero, NY entries from the array Y will be switched corresponding to the switch performed in the X array.

When the sort is complete or IMØDE is 3, control transfers to statement 1052. Following statement 1052 the maximum (XMAX) and minimum (XMIN) entries in array X are found. If NY is less than or equal to zero, control returns to the calling program. Otherwise the maximum (YMAX) and minimum (YMIN) entries for the array Y are calculated. Control then returns to the calling program.


SUBROUTINE SORTR(X,Y,NX,MAXMX,NY,MODE)
N = NX**2
1010 DO 1050 K=1,N
LI = K + 1
LL = NX
MARK = K
1012 DO 1042 L=LI,LL
GO TO (1020,1030,1052),MODE
1020 IF(X(L)-X(MARK)) 1042,1042,1041
1030 IF(X(L)-X(MARK)) 1041,1042,1042
1041 MARK = L
1042 CONTINUE
Calculate number of sorts
DO over array elements
Calculate lower index for sort
Calculate upper index for sort
Store initial array index
DO over unsorted array elements
Branch for specified mode
Is the current array element greater than the stored maximum?
Is the current array element less than the stored minimum?
Reset array maximum (or minimum) element index

P.4.SØRTR-1


Temporarily store extreme (maximum or minimum)
Switch locations of extreme and lower index element
Is the number of elements to be switched from the Y array less than zero?
DO over elements of Y array
Switch locations of elements in Y array corresponding to X array switch
Find maximum element in X array
Find minimum element in X array
Is the number of elements to be switched from the Y array less than zero?
2
1049
SWITCH = XIMARK)
XIMARK) = XIK)
XIK) = SWITCH
IF(NY.LE.0)
F
T
GO TO 1050
GO 1051 I=1,NY
SWITCH = YIMARK.I)
YIMARK.I) = YIK.I)
1051
YIK.I) = SWITCH
1050
CONTINUE
A1
1052
CALL MAXNI(X,NX,XMAX)
CALL MINNI(X,NX,XMIN)
IF(NY.LE.0)
RETURN


A2
DO 1060 I=1,NY
CALL MAXN(Y(1,I),NX,YMX)
CALL MINN(Y(1,I),NX,YMN)
IF(I.EQ.1.OR.YMX.GT.YMAX)
T
F
YMAX = YMX
IF(I.EQ.1.OR.YMN.LT.YMIN)
T
F
YMIN = YMN
1060 CONTINUE
RETURN
END
DO over columns of Y array
Find maximum element for column I
Find minimum element for column I
Is this the first column or has a new maximum been determined?
Set new Y maximum
Is this the first column or has a minimum been determined?
Set new Y minimum

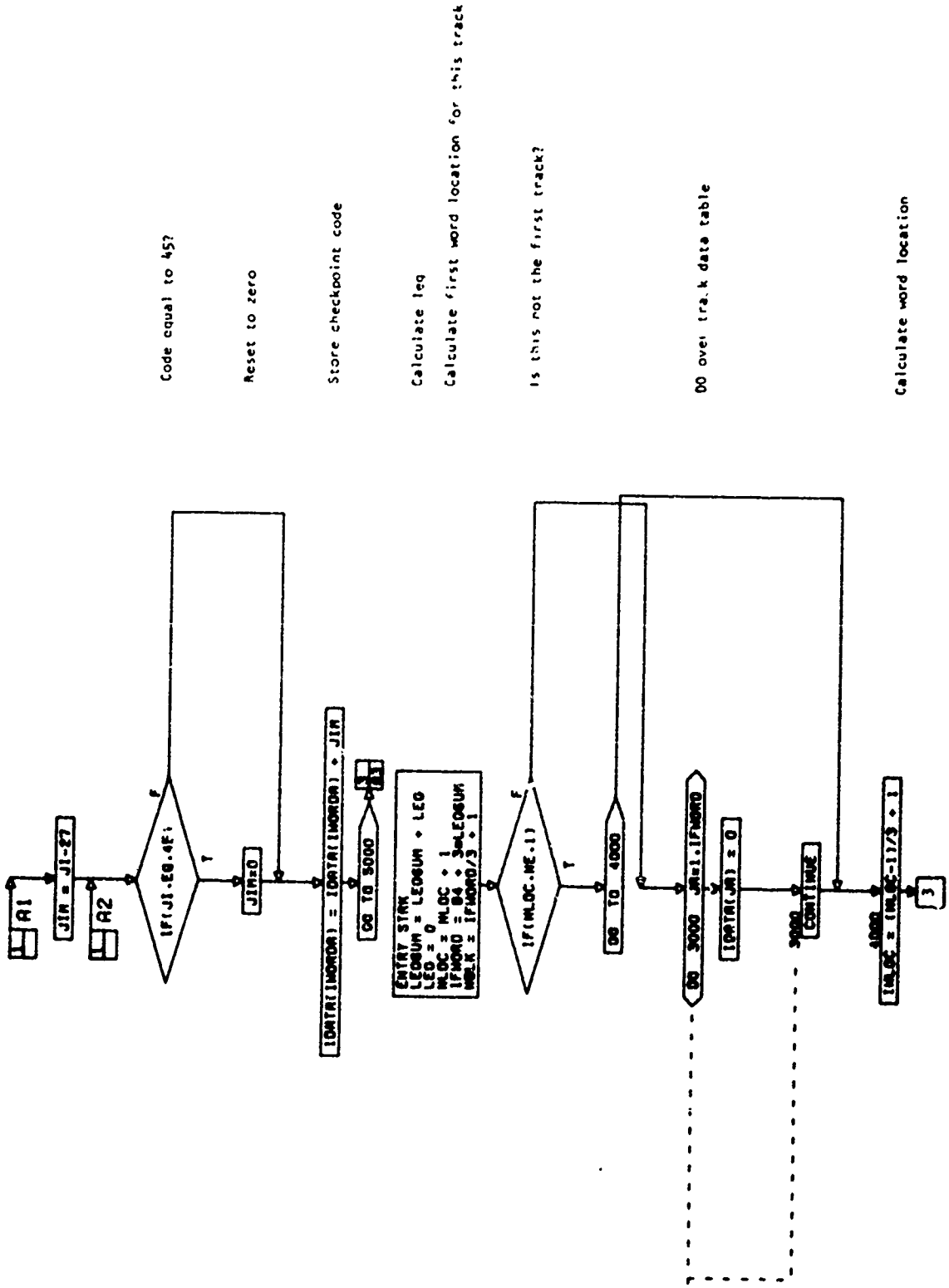
1. NAME: STPNT (other entry points: STRK)
2. TYPE OF PROGRAM: SUBROUTINE, STORAGE
3. LANGUAGE: FORTRAN EXTENDED
4. PURPOSE: Store track data in the track data table.
5. ASSUMPTIONS AND LIMITATIONS: STRK must be called to calculate track point initial address prior to point storage.
6. ERROR RETURNS: A error message will be printed and a RETURN E1 executed if track data table overflows.
7. LINKAGE STATEMENT AND DESCRIPTION OF LINKAGES:
   CALL STPNT (X, Y, Z, T, J), RETURNS (E1)
   entry point CALL STRK (DUM, DUM, DUM, DUM,NDUM), RETURNS (E1)
   X - Relative easting of the track point under consideration.
   Y - Relative northing of the track point under consideration.
   Z - Altitude of the track point under consideration.
   T - Time required to reach the point under consideration.
   J - Check point code for the point under consideration.
   DUM - Dummy variable not used.
   NDUM - Dummy variable not used.
   E1 - Exit taken if the condition listed under 6 (above) occurs.
8. PROGRAMS CALLING THIS PROGRAM:
   entry point STPNT: MAIN
   entry point STRK: MAIN
9. COMMON VARIABLES USED:
   A. ZZTRKD - NDWØRD, TKDATA (equivalenced to IDATA)
10. COMMON VARIABLES TO BE SET:
    A. ZZTRKD - NDWØRD
11. COMMON VARIABLES CHANGED:
    A. ZZTRKD - TKDATA (equivalenced to IDATA)

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES: None

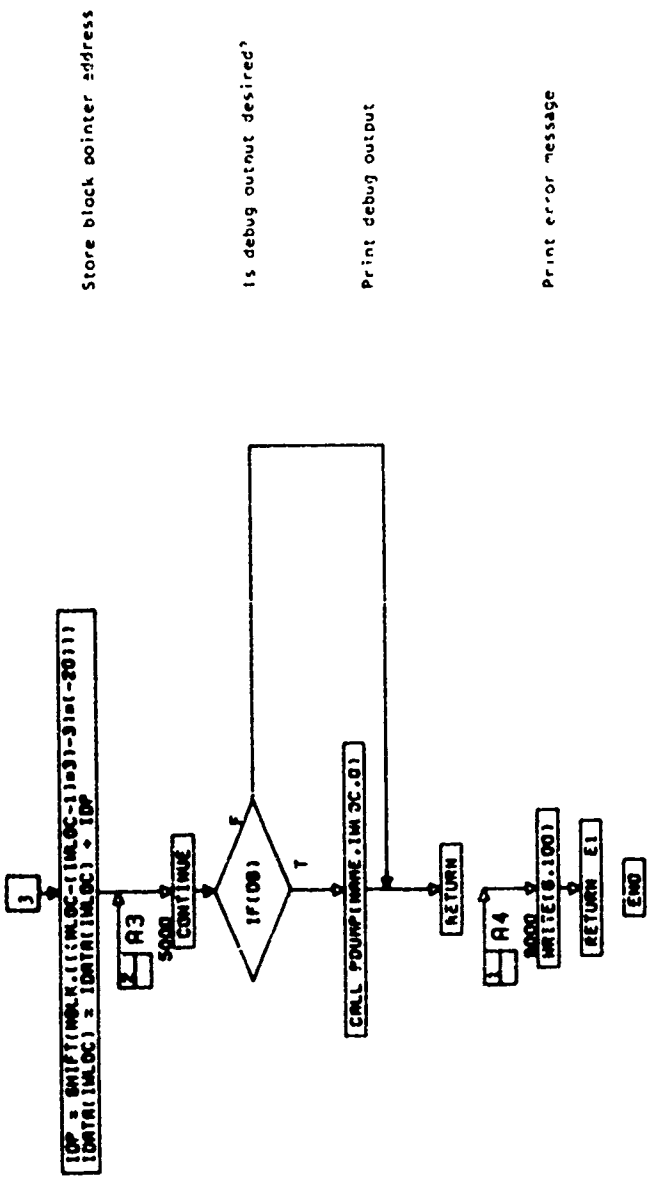
13. NOTES ON METHODOLOGY: Upon entry to STPNT, the path leg index (LEG) is incremented and the address of the first word of storage for this leg is calculated. If this storage area exceeds the maximum word address (NDWORD), control transfers to statement 9000 where an overflow message is printed and control transfers to the calling program via error RETURN E1. If table overflow does not occur, the time to reach this leg endpoint is stored in the track data array (TKDATA) at address IWORDA. The easting of this point is then stored in the first 40 bits of the next word and the and the corresponding track altitude is stored in the last 20 bits of this word. The address is then incremented and the points northing stored in the first 40 bits of the next word and the checkpoint code (MC) stored in the last 20 bits of this word. Control then transfers to statement 5000 where the track array will be dumped if the DEBUG flag (DB) is TRUE. Control is then returned to the calling program.

Upon entry via STRK, the total leg counter (LEGSUM) is incremented by the number of the legs processed for the previous track. The location counter (NLØC) is then incremented by one and the address (IFWORD) of the first word in this block is calculated. Then the block counter (NBLK) is set. If this is the first call to STRK, the track data array is initialized to zero. At statement 4000, the location of the block data pointer address is calculated. The block data address is then stored in the data array. Control then returns to the calling program.

Increment leg counter
Calculate first word location
Has the track data table overflowed?
Store time in track data table
Increment word index
Store point passing in track data table
Is th < point's altitude other than zero?
Determine sign
Store altitude in track data table
Checkpoint code greater than 26?
SUBROUTINE STPNT(X,Y,Z,T,JJ,RETURNS(E1))
LEO = LEO + 1
INWORD = LEO*3 - 2 + IFWORD
IF(INWORD.GT.NOWORDS)
T
F
GO TO 9000
TRDATA(INWORD+1) = T
INWORD = INWORD+ 1
IOP = X
IDATA(INWORD) = Q
IDATA(INWORD) = IDATA(INWORD) + SHIFT(IOP,20)
SIGN = 1.0
IF(Z .NE. 0.0)
SIGN = Z/ABS(Z)
IOP = Z*10.0 + .05*SIGN
IDATA(INWORD) = IDATA(INWORD) + MAG(MAGNZ,IOP)
INWORD = INWORD + 1
IDATA(INWORD) = 0
IOP = Y
IDATA(INWORD) = IDATA(INWORD) + SHIFT(IOP,20)
JJ=SHIFT(MAG(MAGNZ,JJ),-8)
JJ = JJ + 8
IF(JJ.GT.26)


Code equal to 45?
Reset to zero
Store checkpoint code
Calculate leq
Calculate first word location for this track
Is this not the first track?
DO over track data table
Calculate word location
A1
JIN = J1-27
A2
IF(J1.EQ.45)
F
T
J1=0
IDATA(1WORD) = IDATA(1WORD) + J1N
GO TO 9000
ENTRY STRK
LEQSUM = LEQSUM + LEQ
LEQ = 0
MLOC = MLOC + 1
IFWORD = 84 + 3*LEQSUM
WORK = IFWORD/3 + 1
IF(MLOC.NE.1)
GO TO 4000
DO 3000 JA=1,IFWORD
IDATA(JA) = 0
3000 CONTINUE
4000 LMLOC = (MLOC-1)/3 + 1
3

P.4.STPNT-2


Store block pointer address
Is debug output desired?
Print debug output
Print error message
3
IDP = SHIFT(MAX.((IWLOC-(IWLOC-1)*3)-3)*(-20)))
(CONTAI(IWLOC)) + IDP
A3
5000
CONTINUE
IF(DB)
T
CALL POUMP(NAME,IW,3C,01)
RETURN
A4
9000
WRITE(6,100)
RETURN E1
END

1. NAME: THYME (other entry points: TZT)

2. TYPE OF PROGRAM: SUBROUTINE, MATHEMATICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Calculate and print the program accumulated time and incremental time since the last time printout via THYME.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   CALL THYME no arguments

   entry point CALL TZT no arguments

8. PROGRAMS CALLING THIS PROGRAM:

   entry point THYME:

   entry point TZT: MAIN, DUPINP, PRØDMF, PSIT, PSYS

9. COMMON VARIABLES USED: None

10. COMMON VARIABLES SET: None

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

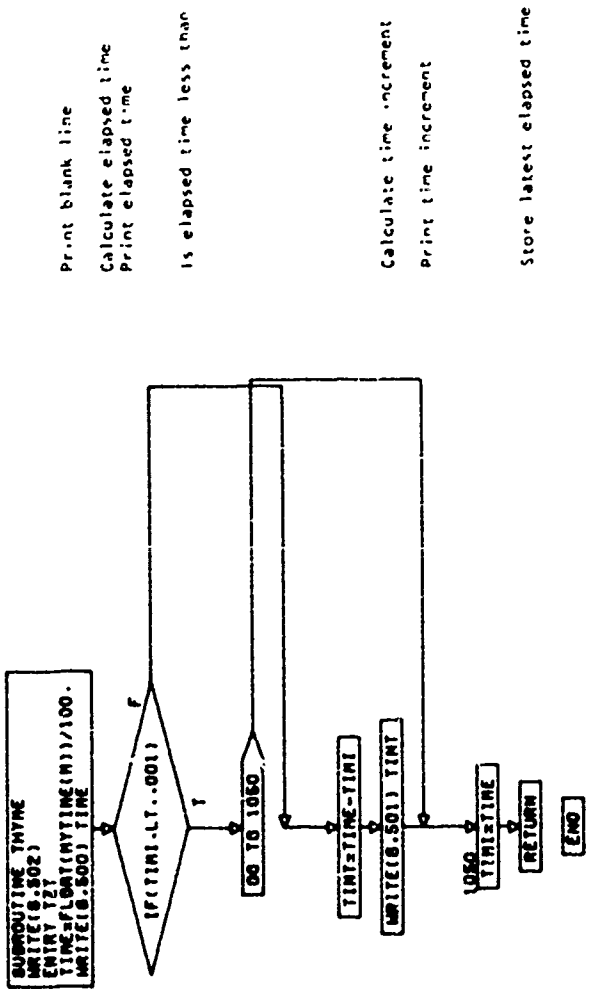
    A. FUNCTIONS

       (1) TIME = MYTIME(M). MYTIME is used to calculate the accumulated CP time in units of 100ths of a second.

           M - Integer, dummy variable not used.

13. NOTES ON METHODOLOGY: Subroutine THYME is used to calculate accumulated CP time and incremental CP times and print a two line time report. Upon entry to THYME a printer line is skipped. Entry via ENTRY TZT first calculates TIME, the accumulated CP time in seconds,

using function MYTIME. A printout of the accumulated time, TIME, is then provided. If ITIM, the accumulated time (stored from previous call), is less than .001 seconds control transfers to statement 1050 where TIMI is set to the current accumulated time and a normal exit to the calling subroutine taken. If ITIM is greater than or equal to .001, the incremental CP time is calculated from the difference between the current time, TIME, and previous time, ITIM. A printout of this incremental time is provided, ITIM is set to TIME and control is returned to the calling program.


SUBROUTINE THYME
WRITE(6,502)
ENTRY TZY
TIME=FLOAT(MYTIME(N))/100.
WRITE(6,500) TIME
IF(TIME.LT..001)
F
T
GO TO 1060
TINI=TIME-TINI
WRITE(6,501) TINI
1060
TINI=TIME
RETURN
END
Print blank line
Calculate elapsed time
Print elapsed time
Is elapsed time less than .001?
Calculate time increment
Print time increment
Store latest elapsed time

1. NAME: TTØRNG
2. TYPE OF PROGRAM: SUBROUTINE, MATHEMATICAL
3. LANGUAGE: FORTRAN EXTENDED
4. PURPOSE: Calculate the time a specified track enters a specified site fire volume.
5. ASSUMPTIONS AND LIMITATIONS: A entry is recorded only if this time is after TIZZ.
6. ERROR RETURNS: None
7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:
   CALL TTØRNG (RAGIN, TIZZ, TIME, NF, NTK)
   RAGIN - Range of interest for the site under consideration.
   TIZZ - Earliest time to be considered for entry.
   TIME - Time a specified track enters a specified site fire volume.
   NF - Integer, index of the site under consideration.
   NTK - Integer, index of the track under consideration.
8. PROGRAMS CALLING THIS PROGRAM: MAIN
9. COMMON VARIABLES USED:
   A. ZZTRK1 - MNP
   B. ZZSITE - XFU, YFU, ZFU
10. COMMON VARIABLES TO BE SET:
    A. ZZTRK1 - MNP
    B. ZZSITE - XFU, YFU, ZFU
11. COMMON VARIABLES CHANGED: None
12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:
    A. FUNCTIONS
       (1) X = TTRK (NTK,I) - Finds the time for the specified track and point indices.

NTK - Index for the specified track.

I - Index for the specified point.

(2) X = XTRK (NTK, I)(entry point to TTRK) - Finds the easting for the specified track and point indices.

NTK - Index for the specified track.

I - Index for the specified point.

(3) X = YTRK (NTR, I) (entry point to TTRK)- Finds the northing for the specified track and point indices.

NTK - Index for the specified track.

I - Index for the specified point.

(4) X = ZTRK (NTK, I) (entry point to TTRK) - Finds the altitude for the specified track and point indices.

NTK - Index for the specified track.

I - Index for the specified point.

13. NOTES ON METHODOLOGY: Upon entry to TTØRNG, the site location is stored in local variables XFUT, YFUT, AND ZFUT. The square of the site range of interest is calculated, the number of points on the track is stored locally, and the point counter (I) for this track is initialized to 1.

At statement 1005, a loop is entered which finds the time of intersection for a specified track with a specified site fire volume. First, the point counter is indexed and a test performed to see if the maximum number of points for this track has been exceeded. If the maximum number of points have been exceeded (I>MN), control transfers to statement 1036 where the time of intersection is set equal to -1 and control returns to the calling program. If the number of points has not been exceeded, a test is performed to see if this leg end point is reached before the specified initial time (TIZZ). If this leg end point is reached by the

vehicle before the specified initial time, control transfers to statement 1005 where the point index is incremented in preparation for the next leg to be tested. Next, the square of the range from the site location to the leg starting point and end point (ANS,CNS) are calculated. Time is set equal to the leg initialization time and the range to the starting point is tested. If this range to the leg starting point is less than the range of interest, control returns to the calling program. If this point lies outside the range of interest, the leg extension variable DISCRM is calculated. If DISCRM is less than zero, this leg does not intersect the fire volume and control transfers to statement 1005. If this leg intersects the fire volume, the entry time and exit time for this leg (or leg extension) are calculated. If this time satisfies the desired constraints, the entry time is stored in TIME and control then transfers to the calling program.

Store site location

Square site range of interest

Initialize leg counter

Increment leg counter

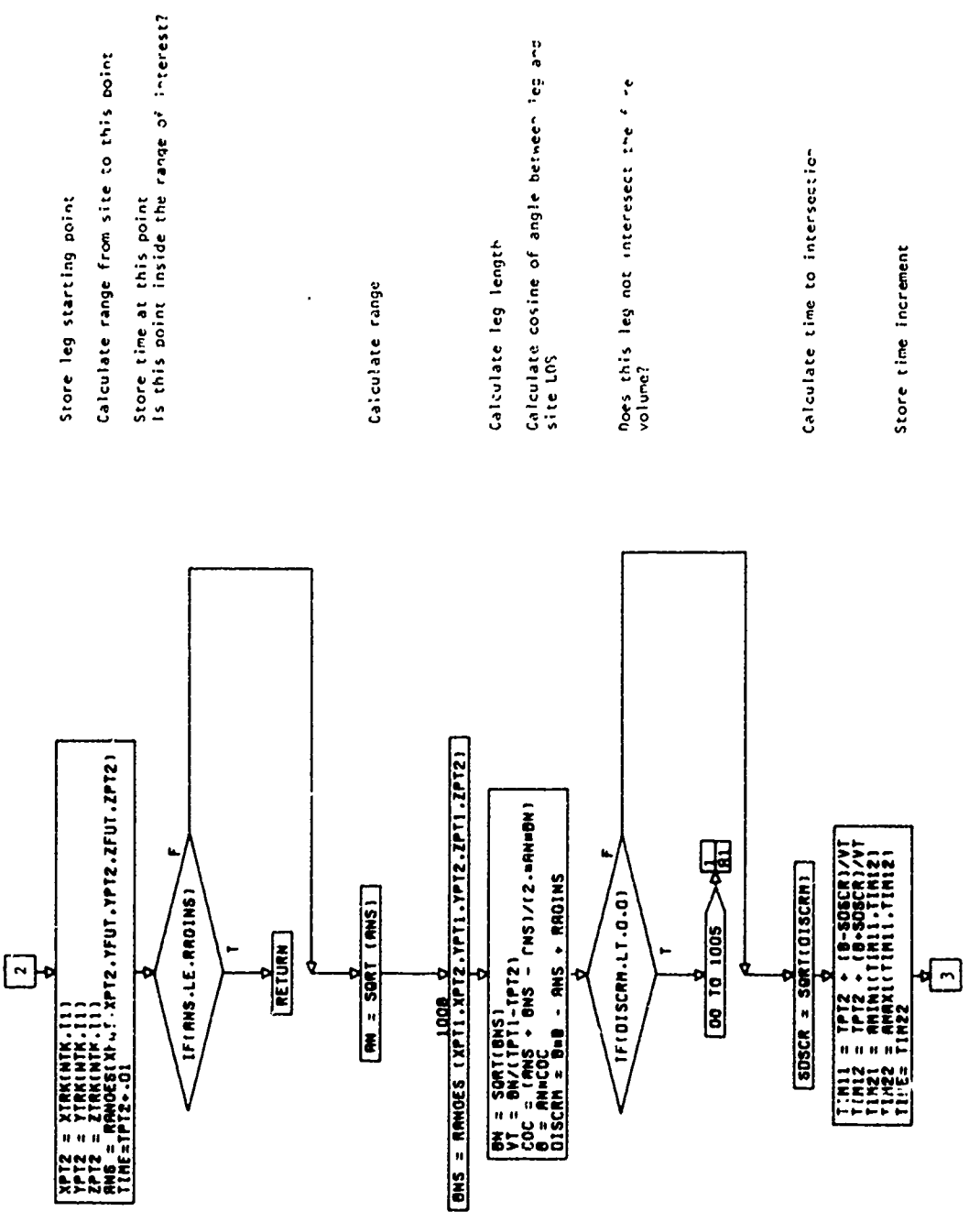
Have all the legs been checked for an intersection?

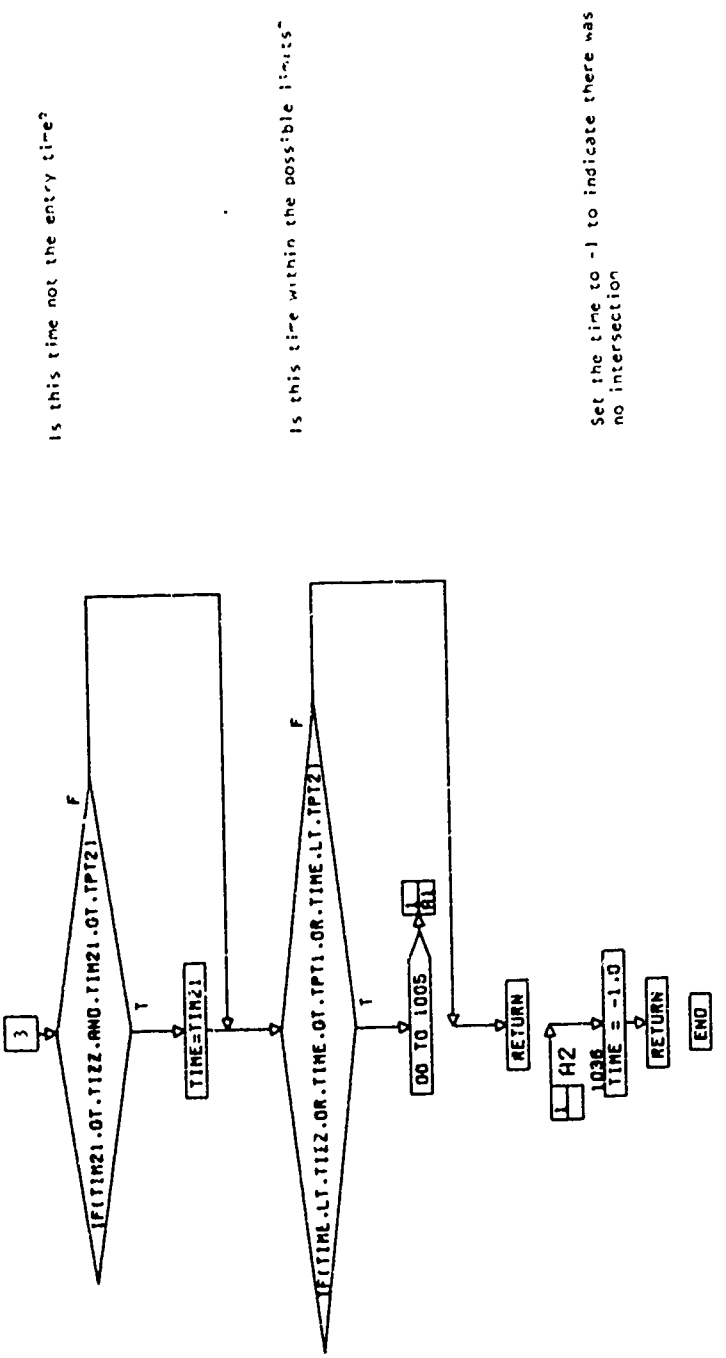
Is this leg traversed before the initial time of interest?

Store leg end point

Calculate range from site to this point

Decrement leg counter
Store time


Store leg starting point
Calculate range from site to this point
Store time at this point
Is this point inside the range of interest?
Calculate range
Calculate leg length
Calculate cosine of angle between leg and site LOS
Does this leg not intersect the ' re volume?
Calculate time to intersection
Store time increment
2
XPT2 = XTRK(NTK,I1)
YPT2 = YTRK(NTK,I1)
ZPT2 = ZTRK(NTK,I1)
RNG = RANGES(XPT2,YPT2,ZPT2,XFUT,YFUT,ZFUT)
TIME=TPT2+.01
IF(ANS.LE.RADIUS)
F
T
RETURN
RN = SQRT (ANS)
1008
BNS = RNGES (XPT1,XPT2,YPT1,YPT2,ZPT1,ZPT2)
BN = SQRT(BNS)
VT = BN/(TPT1-TPT2)
COC = (ANS + BNS - RNS1)/(2.*RN*BN)
B = ANCOC
DISCRM = B*B - ANS + RADIUS
IF(DISCRM.LT.0.0)
F
T
DO TO 1005
81
SDSCR = SQRT(DISCRM)
TIM11 = TPT2 + (B-SDSCR)/VT
TIM12 = TPT2 + (B+SDSCR)/VT
TIM21 = AMIN1(TIM11,TIM12)
TIM22 = AMAX1(TIM11,TIM12)
TIME= TIM22
3


Is this time not the entry time?
Is this time within the possible limits?
Set the time to -1 to indicate there was no intersection
3
IF(TIM21.GT.TIZZ.AND.TIM21.GT.TPT2)
F
T
TIME=TIM21
IF(TIME.LT.TIIZ.OR.TIME.GT.TPT1.OR.TIME.LT.TPT2)
F
T
GO TO 1005
81
RETURN
A2
1036
TIME = -1.0
RETURN
END

1. NAME: TTRK (other entry points: XTRK, YTRK, ZTRK)

2. TYPE OF PROGRAM: FUNCTION, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Find values of time, easting, northing, and altitude from the track array.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:
   entry point X = TTRK(NTK,I)
   entry point X = XTRK(NTK,I)
   entry point X = YTRK(NTK,I)
   entry point X = ZTRK(NTK,I)
   NTK - Integer, index of the path of interest.
   I - Integer, index of the point of interest.

8. PROGRAMS CALLING THIS PROGRAM:
   TTRK: MAIN, TTØRNG
   entry point XTRK: MAIN, TTØRNG
   entry point YTRK: MAIN TTØRNG
   entry point ZTRK: MAIN, TTØRNG

9. COMMON VARIABLES USED:
   A. ZZTRK1 - MNP
   B. ZZSITE - XFU, YFU, ZFU

10. COMMON VARIABLES TO BE SET:
    A. ZZTRK1 - MNP
    B. ZZSITE - ZFU, YFU, ZFU

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:

A. SUBROUTINES

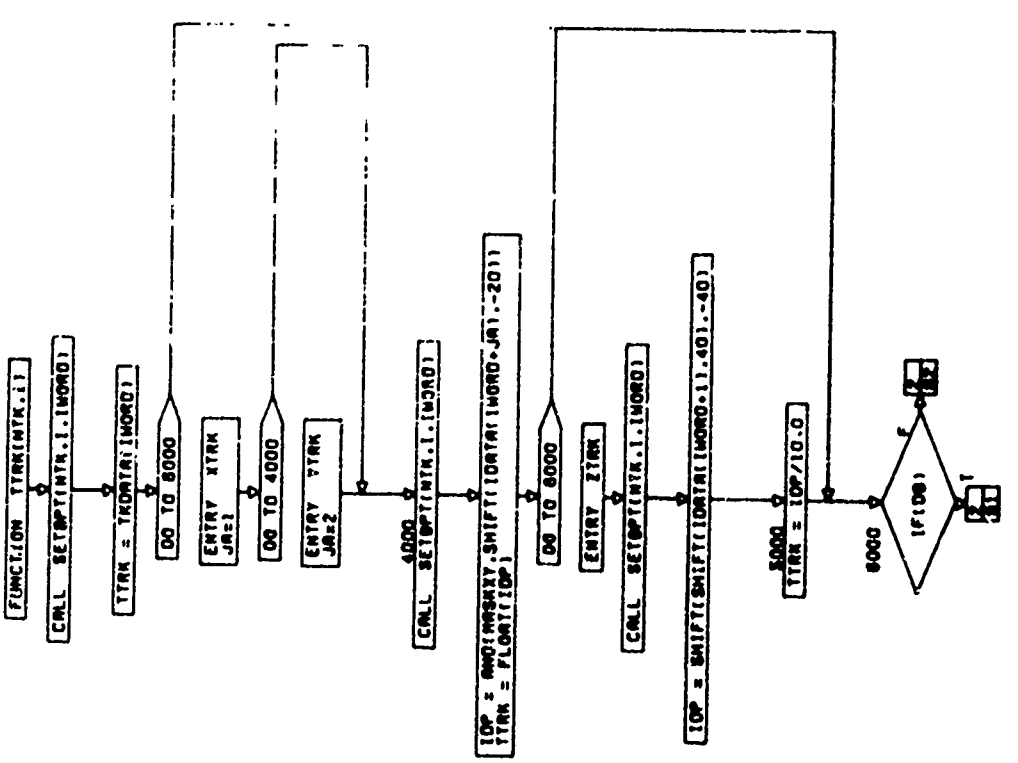
CALL SETBPT (NTK,I,IWØRD)

NTK - Index of the desired track.

I - Index of the desired point.

IWØRD - Block pointer address for the desired point.

13. NOTES ON METHODOLOGY: Function TTRK with entry points XTRK, YTRK, and ZTRK, is used to find the time, easting and northing for a specified track point. Upon entry via TTRK, the block pointer word (IWØRD) is calculated by a call to subroutine SETBPT. The time is then stored in variable TTRK and control transfers to statement 6000. If entry is via entry point XTRK, the address index (JA) is initialized to 1 and control transfers to statement 4000. If entry is via entry point YTRK, the address index is initialized to 2 and control transfers to statement 4000. At statement 4000, SETBPT is called to calculate the block pointer address for this track point. The easting (JA=1) or northing (JA=2) is then stored and control transfers to statement 6000. If entry is via entry point ZTRK, SETBPT is called to find the block pointer address for this point. The altitude at this point is then calculated at statement 5000. The debug flag (DB) is then tested. If the debug flag is TRUE, PDUMP is called to dump the track data array and control returns to the calling program.



Find block address

Store time required for threat to reach this track point

Set easting word index

Set northing word index

Find block address

Store easting or northing from track data array

Find block address

Store altitude of this point from array

Is debug output desired?

P .4.TTRK-1

Print debug output

1. NAME: ZCKPT1 (other entry points: ZCKPT2, ZCKPT3, ZCKPT4)

2. TYPE OF PROGRAM: FUNCTION, LOGICAL

3. LANGUAGE: FORTRAN EXTENDED

4. PURPOSE: Set the logical check point flag corresponding to the appropriate bit.

5. ASSUMPTIONS AND LIMITATIONS: None

6. ERROR RETURNS: None

7. LINKAGE STATEMENT AND DESCRIPTION OF ARGUMENTS:

   entry point X = ZCKPT1(NTK,I)
   entry point X = ZCKPT2(NTK,I)
   entry point X = ZCKPT3(NTK,I)
   entry point X = ZCKPT4(NTK,I)

   All use the same arguments.
   NTK - Integer, Index of the path of interest.
   I - Integer, Index of the point of interest.

8. PROGRAMS CALLING THIS PROGRAM: None

9. COMMON VARIABLES USED:
   A. ZZTRKD - TKDATA (equivalenced to IDATA)

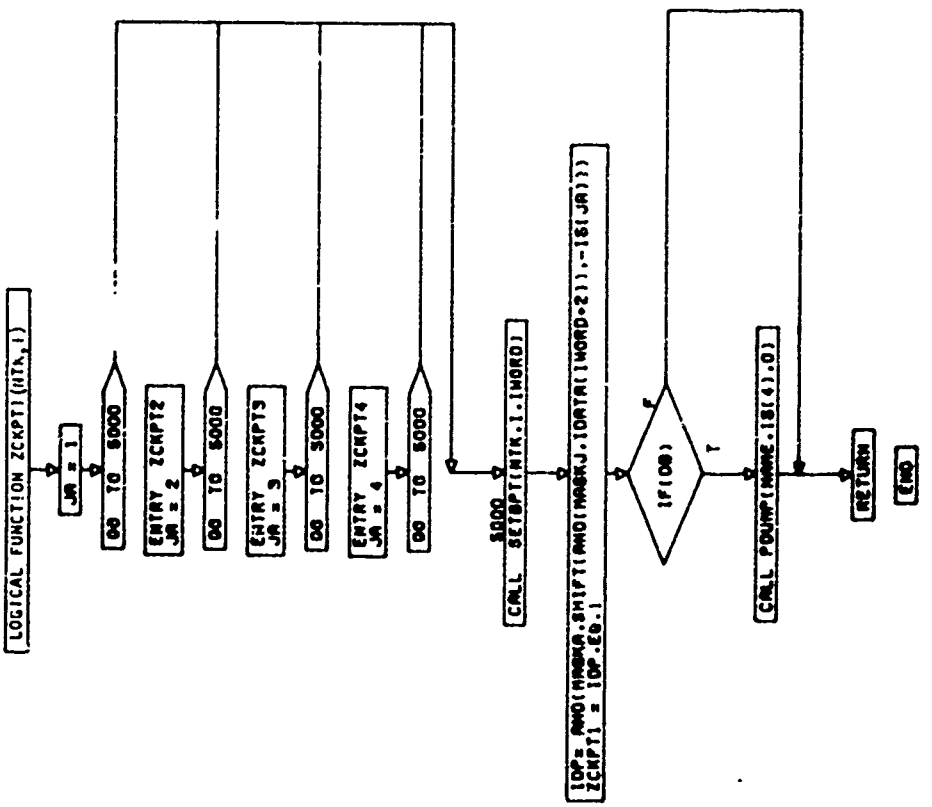
10. COMMON VARIABLES TO BE SET:
    A. ZZTRKD - TKDATA (equivalenced to IDATA)

11. COMMON VARIABLES CHANGED: None

12. PROGRAMS USED AND DESCRIPTION OF LINKAGES:
    A. SUBROUTINES:
       CALL SETBPT (NTK, I, IWØRD)
       NTK - Index of the desired track.
       I - Index of the desired point.
       IWØRD - Block pointer address for the desired point.

13. NOTES ON METHODOLOGY: Entry to this function via entry points ZCKPT1, ZCKPT2, ZCKPT3, and ZCKPT4 specifies the checkpoint bit index for this track point. Upon entry the first, second, third,

or fourth bit of the check point code will be selected for test. At statement 5000, the block pointer address for this track point is calculated. Then the first, second, third, or fourth checkpoint bit is calculated and stored in local variable IDP. Logical checkpoint flag ZCKPTI is then set TRUE if IDP is equal to 1. A test is then performed to determine if debug output is desired and control returns to the calling program.



Set index for a shift of 3 bits

Set index for a shift of 2 bits

Set index for a shift of 1 bit

Set index for a shift of zero bits

Find index into path data storage area where this path/point combination data are stored

Get 1 - bit of information from maneuver code
Set flag as to whether the bit is on or off

Print debug information?

Dump storage locations for this function

P .4.ZCKPT1-1

P.5 ERROR MESSAGE LIST

The following is a error message list for PMAP. To facilitate error lookup, the errors are categorized by the subroutine name and the statement number in the subroutine where the error was detected. Error detection sometimes generates several messages providing a trace to the source of the error. Some of the statement numbers were not properly initialized prior to error message printing. These statement numbers are listed or 'XXXX'.

P.6 OUTPUT RECORD FORMATS

PMAP produces a printed output report and map plot and if requested, a tape consisting of the sites which engage at least one path.

P.6.1 Output Report and Map Plot

PMAP produces a site data table, site plot, path data table, path point plot, and engageability table. Annotated examples of each of these types of output may be found in Volume III D. TERAIN, PMAP, SØRTEV User/Planner Manual, Paragraph P.4.

P.6.2 Site Tape

If requested by the analyst, PMAP will punch one seven word logical record for each site which has at least one engageable path. As shown in Figure P.6-1, this record contains the system identifier, site identifier, site location in UTM coordinates, site altitude, and the site sector.

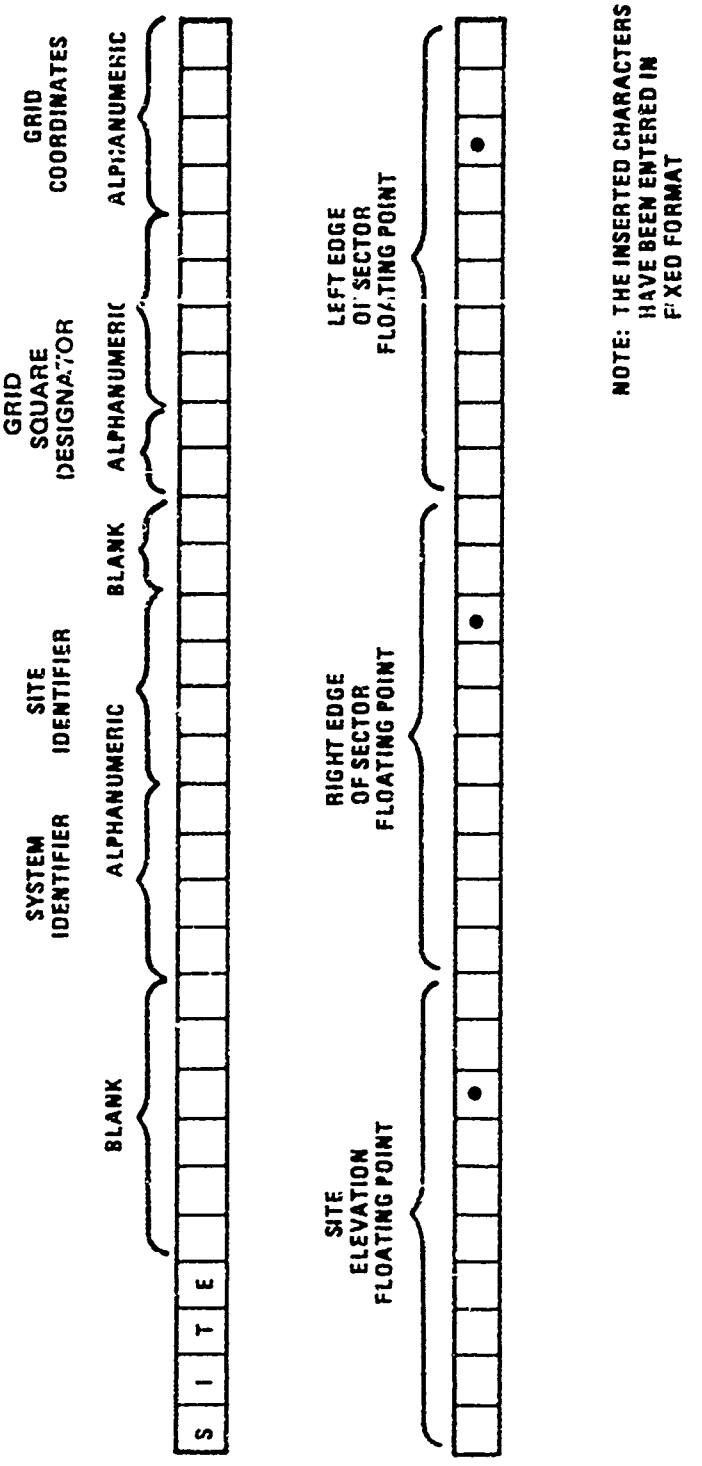


Figure P.6-1. Engageable Site Record

P.7 MACHINE ENVIRONMENT

P.7.1 Hardware Requirements

The MICOM version of TACOS II will run on any computer in the CDC 6000, 7000, or CYBER-70 series.

P.7.1.1 Core Memory

PMAP requires 56175 (octal) words of core memory for execution. The size of the loader at the installation under consideration must be accounted for to obtain the core requirement for loading. ECS is not required.

P.7.1.2 Disk Tape Space

A minimum of two 7-track tape units are required during the execution of PMAP. The precise number of PRU's of disk storage use will depend on the size of the terrain file used. Each 100 KM grid square contains 22 1717 word records and 2 1515 word records or a total of 40,804 words per grid square. Grid zone 32UF contains 12 100 KM grid squares, 489,648 words or 7,651 PRUs.

P.7.1.3 Other I/O Devices

A card reader and line printer are necessary for normal I/O interfacing with the user.

P.7.2 Operating System Requirements

P.7.2.1 Basic System

The operating system for the MICOM version of TACOS II is SCOPE 3.4. The source language for PMAP is compiled under FORTRAN EXTENDED V4.

P.7.2.2 Non-Standard Options

PMAP requires no non-standard options.

P.7.2.3 Operator Instructions

If a non-standard line spacing is specified, that is, other than 6 lines per inch, the operator should be informed prior to program execution.

APPENDIX A

GLOSSARY FOR PMAP

PMAP

VARIABLE	COMMON	TYPE	DESCRIPTION
AZ1A(255)	ZZSITE	REAL	Left edge of site sector (degrees from north).
AZ2A(255)	ZZSITE	REAL	Right edge of site sector (degrees from north).
DELTA	ZZCORD	REAL	Map scale in meters per printed inch.
ESTEDG	ZZCORD	REAL	Eastern edge of the north-south strip under consideration.
IBUF(1717)	ZZBUF	INTEGER	Buffer for reading direct access terrain data.
ID(255)	ZZPLOT	INTEGER	Identifiers to be plotted in locations (X,Y) on a map.
IDFU(255)	ZZSITE	INTEGER	AD site identifier.
IDFUTY(255)	ZZSITE	INTEGER	System identifier for each site.
IDTHV(255)	ZZTRKI	INTEGER	Identifiers of the penetrator type on each path.
IDTRK(255)	ZZTRKI	INTEGER	Identifier of each path.
IMAX	ZZRAI	INTEGER	Maximum number of records of terrain data to be transferred to mass storage.
IORBUF(56)	ZZCORD	INTEGER	Buffer containing the abscissa coordinates for map header printing.
IORG	ZZCVI	INTEGER	Integer form of XORG.
IRAN(290)	ZZRAI	INTEGER	Random access index array.
JORG	ZZCVI	INTEGER	Integer form of YORG
KGZSV	ZZFASC	INTEGER	Four character grid zone identifier for grid zone of interest.
LNBUF(14)	ZZCORD	INTEGER	Line buffer used for storing one line prior to map printing.

PMAP

VARIABLE	COMMON	TYPE	DESCRIPTION
LODA(30)	ZZFASC	INTEGER	Relative track address of first track containing data for the corresponding 100 km square.
LTDA(30)	ZZFASC	INTEGER	Two letter designation of each 100 km square on this grid zone file.
LTORG	ZZCV1	INTEGER	Two letter UTM grid square designator for coordinate system origin.
MNP(255)	ZZTRK1	INTEGER	Number of points on each path.
NAME	F.P.	INTEGER	Name of subroutine (JTRK,SETBPT).
NCL(2)	ZZCV3	INTEGER	Two 4 digit grid coordinater of a map lower left corner in printable format.
NCU(2)	ZZCV3	INTEGER	Two 4 digit grid coordinater of a map upper right corner in printable format.
NDWORD	ZZTRKD	INTEGER	Block pointer index limit (3000).
NLPI	ZZMAPP	INTEGER	Number of lines per inch in output listing.
NODA	ZZFASC	INTEGER	Number of 100 km squares in this direct access terrain file.
SCALE	ZZMAPP	REAL	Map scale in ground meters per printed inch.
TKDATA(3000)	ZZTRKD	REAL	Array containing the packed path point data (easting, northing, altitude, MC).
WSTEDG	ZZCORD	REAL	Western edge of the north-south strip under consideration.
X(255)	ZZPLOT	REAL	Easting of either path points or site locations to be plotted on a map.
XFU(255)	ZZSITE	REAL	Easting of each AD site.
XMAX	ZZMAPP	REAL	Easting of the western-most point to be plotted.

PMAP

VARIABLE	COMMON	TYPE	DESCRIPTION
XMIN	ZZMAPP	REAL	Easting of the eastern-most point to be plotted.
XORG	ZZCV1	REAL	Easting of the coordinate system origin relative to AA.
Y(255)	ZZPLOT	REAL	Northing of either path points or site locations to be plotted on a map.
YFU(255)	ZZSITE	REAL	Northing of each AD site.
YMAX	ZZMAPP	REAL	Northing of the northern-most point to be plotted.
YMIN	ZZMAPP	REAL	Northing of the southern-most point to be plotted.
YORG	ZZCV1	REAL	Northing of the coordinate system origin relative to AA.
ZFU(255)	ZZSITE	REAL	Altitude of the AD site (above MSL).